Proceedings of the

4th International Symposium on Imitation in Animals and Artifacts

Edited by:

José Santos-Victor, Manuel Lopes, Alexandre Bernardino

Newcastle upon Tyne, April 2007.

Table of Contents

4th International Symposium on Imitation in Animals and Artifacts

Preface:

Imitation facilitates transmitting culture practices and ideas from generation to generation, enabling humans, animals, and now robots, to learn skills others have already mastered. By avoiding the lengthy period of trial-and-error to accomplish new tasks, imitation is thus a very efficient learning method, and also a very intuitive way to program robots by teaching.

The mechanisms of imitation and social learning are not well-understood, and the connections to social interaction, communication, development, and learning are deep, as recent research from various disciplines has started to uncover. Comparison of imitation in animals and artifacts reveals that easy tasks for machines can be hard tasks for animals and vice-versa. However, computational complexity issues do not explain, by themselves, the existence or not of imitation behaviours in animals, and the integration of higher level cognitive capabilities like agent's goals, intentions and emotions, may play a fundamental role in explaining these differences.

This interdisciplinary workshop will bring together researchers from neuroscience, brain imaging, animal psychology, computer science and robotics to examine the latest advances to imitation, aiming to further advance our understanding of the underlying mechanisms. We hope that the workshop will contribute to the advance in research in imitation and a better integration between the several scientific disciplines.

The symposium will consist of invited talks, regular presentations and short presentations. It is our privilege to have three distinguished invited speakers: Marcel Brass from the Ghent University, Belgium, speaking on the neuronal mechanisms of imitation and Nicola McGuigan from the Heriot-Watt University, Scotland speaking on imitation in children.

We would like to acknowledge the financial support provided by euCognition – the European Network for the Advancement of Artificial Cognitive Systems and by the EU-Project RobotCub, to all the people who have contributed towards making this symposium happen, the programme review committee and, above all, the authors and participants.

José Santos-Victor, Manuel Lopes, Alexandre Bernardino (symposium chairs)

Instituto Superior Técnico, Lisboa, Portugal
http://vislab.isr.ist.utl.pt

Newcastle upon Tyne, April 2007.

Support:

Symposium Topics:

* Cognitive Development and Imitation
* Neurobiological Foundations of Imitation
* Social interaction and Imitation
* Language acquisition
* Imitation, Intentionality and Communication
* Imitation in Animals
* Learning by Imitation to bootstrap the acquisition of skills & knowledge
* The Role of Imitation in the Development of Social Cognition
* Robot Imitation
* Computational mechanisms of imitation
* Joint-attention and perspective taking
* Cultural transmission of skills
* Teaching and scaffolding of behaviours
* Imitation and motivation

Programme Committee:

Alexandre Bernardino, (IST, Portugal)
Andrew Meltzoff, (U. Washington, USA)
Aris Alissandraiks, (Hertfordshire, UK)
Aude Billard, (EPFL, CH)
Bart Jansen, (VUB, Belgium)
Brian Scassellati, (Yale, USA)
Chana Akins, (Kentucky, USA)
C. L. Nehaniv, (Hertfordshire, UK)
Frédéric Kaplan, (EPFL, Switzerland)
Francisco Lacerda, (Stockholm, Sweden)
Giorgio Metta, (Genova, IT)
Harold Bekkering, (Nijmegen, NL)
Heiko Wersing, (Honda R.I., Germany)
Irene Pepperberg, (Harvard, USA)
Jacqueline Nadel, (CNRS, France)
Jochen J. Steil, (Bielefeld, Germany)
Joanna Bryson, (Bath, UK)
José Santos-Victor, (IST, Portugal)
K. Dautenhahn, (Hertfordshire, UK)
Ludwig Huber, (Vienna, Austria)
Manuel Lopes, (IST, Portugal)
Max Lungarella, (Tokyo, JP)
Monica Nicolescu, (Nevada, USA)
Nicola McGuigan, (St. Andrews, Scotland)
Rui Prada, (IST, Portugal)
Thomas R. Zentall, (Kentucky, USA)
Tony Belpaeme,(Plymouth, UK)
Yukie Nagai, (Bielefeld, Germany)
Yiannis Demiris,(Imperial College, UK)

Invited Talks

Imitation and self/other distinction

Marcel Brass
Department of Experimental Psychology, Ghent University, Ghent, Belgium

There is converging evidence from different fields of cognitive neuroscience suggesting that the observation of an action leads to a direct activation of an internal motor representation in the observer. It has been argued that these shared representations form the basis for imitation, action understanding and mentalizing. However, if there is a shared representational system of perception and action, the question arises how we are able to distinguish between intentionally formed motor representations and externally triggered motor plans. I will first outline empirical evidence and theoretical accounts supporting the idea of shared representations. Then I will review neurological data as well as data from social psychology and cognitive neuroscience suggesting that self/other distinction is a crucial requirement of a shared representational system. Finally, I will present recent findings showing that the mechanisms involved in the control of shared representations share neural resources with social cognitive abilities such as action understanding and mentalizing. Taken together, these data point to the fundamental role of self/other distinction in social cognition.

Imitation of causally-opaque versus causally-transparent tool use by 3- and 5-year-old children.

Nicola McGuigan[1], Andrew Whiten[2], Emma Flynn[2], and Victoria Horner[2]
[1] School of Life Sciences, John Muir Building, Heriot Watt University, Edinburgh, EH4 4AS.
[2] School of Psychology, University of St Andrews, St Andrews, Fife, KY16 9JP.

We explored whether the tendency to imitate or emulate is influenced by the availability of causal information, or the amount of information available in a display. Three and five-year-old children were shown how to obtain a reward from either a clear or an opaque puzzle-box by a live or video model. Each demonstration involved two different types of actions. The first stage involved causally irrelevant actions and the second stage involved causally relevant actions. When presented with the clear box it could clearly be seen that the actions were irrelevant as the causal information was available. In contrast this information was not available with the opaque box, potentially making discrimination between irrelevant and relevant actions difficult. We predicted that the 3-year-olds would imitate with both boxes, whereas the greater cognitive sophistication and causal understanding of the 5-year-olds would allow them to switch between imitation and emulation depending on the availability of causal information. However, the results showed that both 3-and 5-year-old children imitated the irrelevant actions regardless of the availability of causal information following a live demonstration. In contrast the 3-year-olds employed a more emulative approach when the information available in the display was degraded via a video demonstration containing the puzzle box and the actions of the model only. The results indicated that the 5-year-olds were unaffected by the degraded information and continued to employ an imitative approach. We suggest that imitation is such an adaptive human strategy that it is often employed at the expense of efficiency.

Regular Papers

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.

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

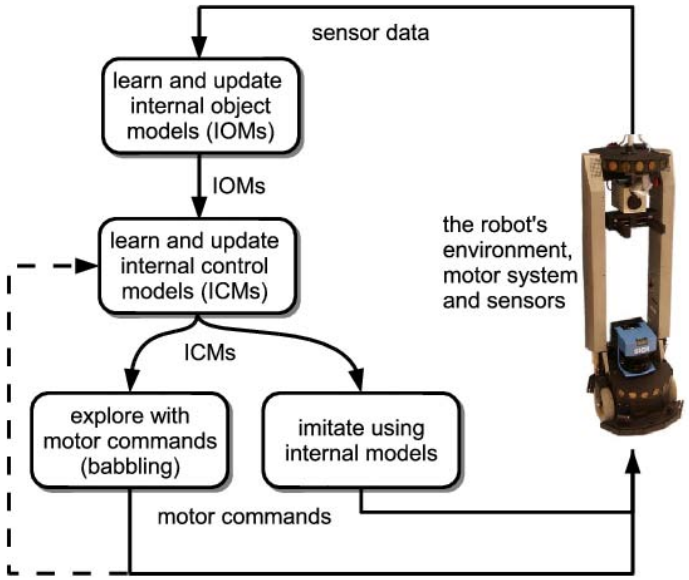


Figure 1. Overview of the learning software.

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.

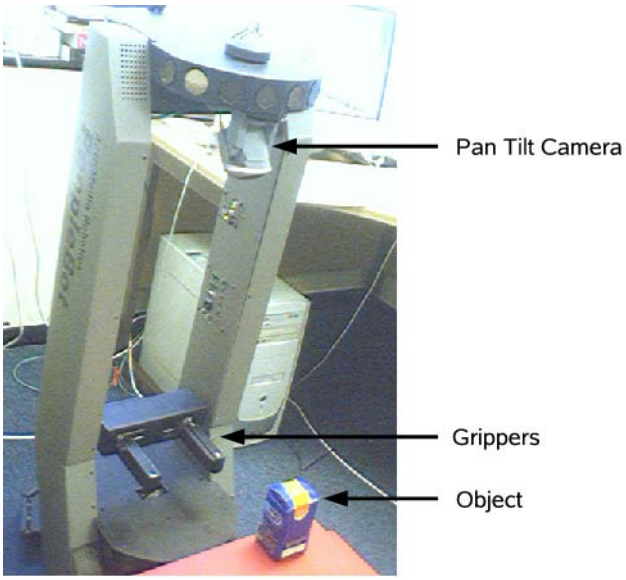


Figure 2. The experimental setup.

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.

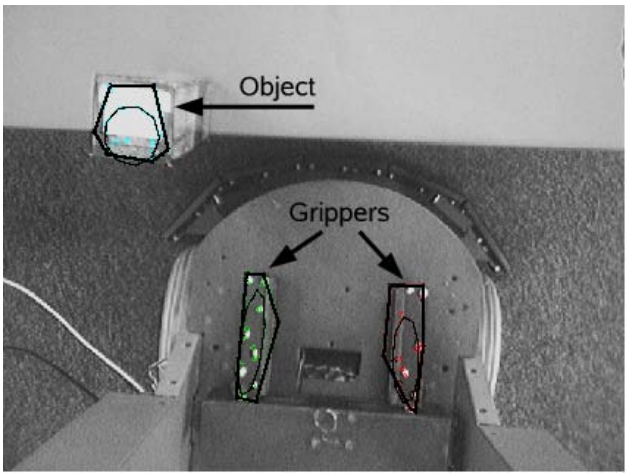


Figure 3. Moving image regions are clustered together; these regions are the robot's IOMs - internal models of where objects are in the scene. In this example, the grippers were moved by the robot, and the biscuit box object was shaken by a human demonstrator to make the robot aware of it. The thick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

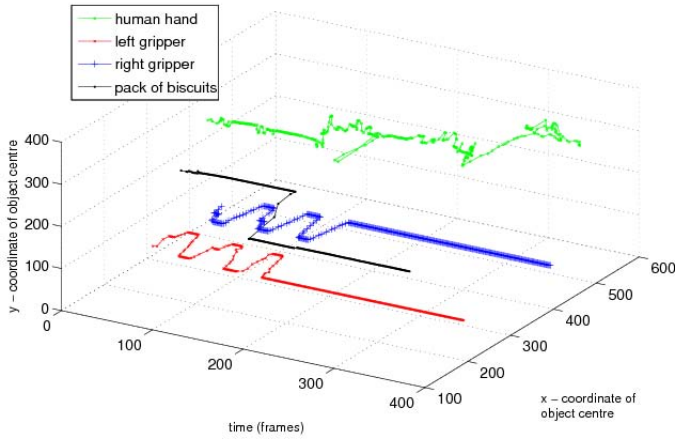


Figure 4. The movement of the IOMs in an experiment, as the grippers open and close and a human hand pushes a box of biscuits.

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.

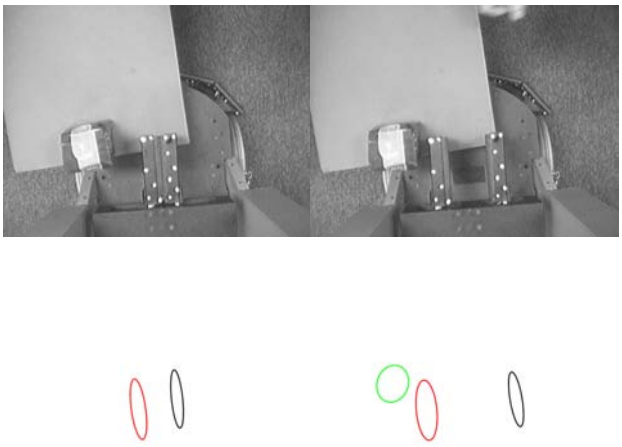


Figure 5. The robot can discover objects by moving them with its own motor system. The top images show frames from the robot 'babbling' in the environment. The bottom frames show the IOMs the robot has discovered before and after the movement.

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

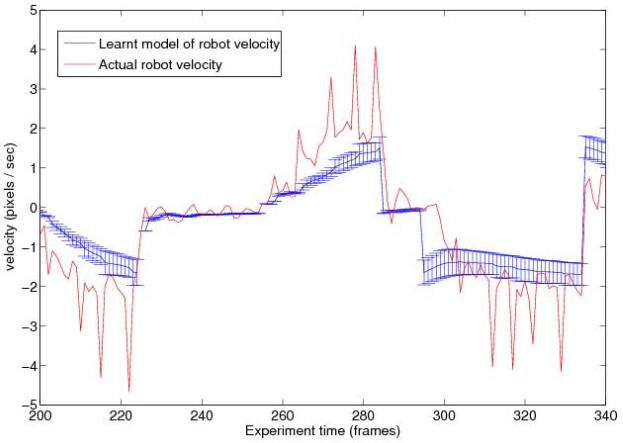


Figure 6. The robot learns online the mean and variance (shown with the error bars) of its velocity as it 'babbles' forwards and backwards. This is the prediction from the most accurate model, for which d=5. The large spikes in the actual data are because of dropped frames from the camera; the robot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

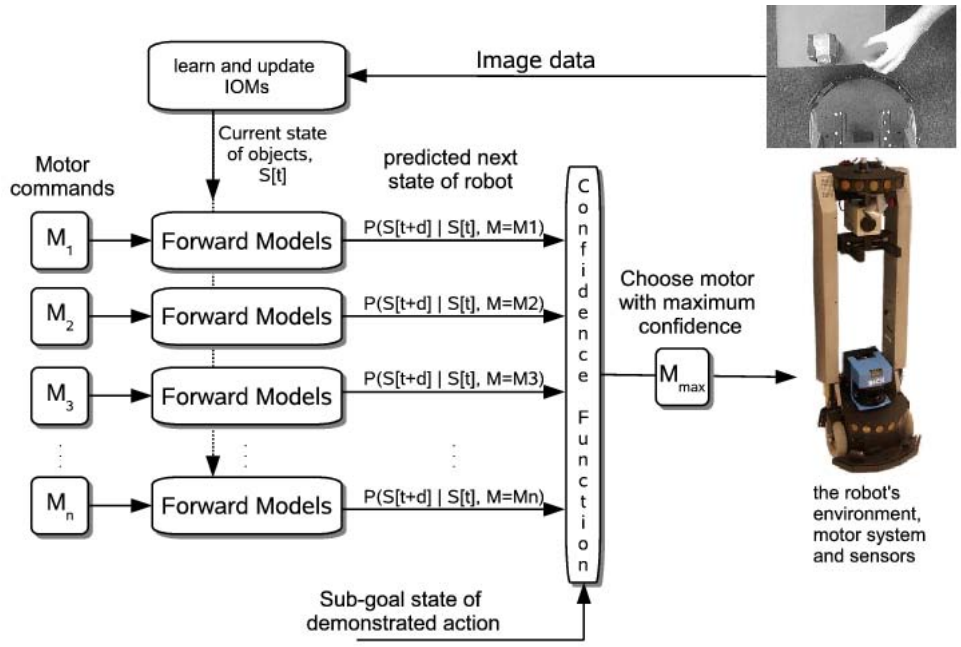


Figure 9. The imitation architecture, using internal models learnt through exploration.

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

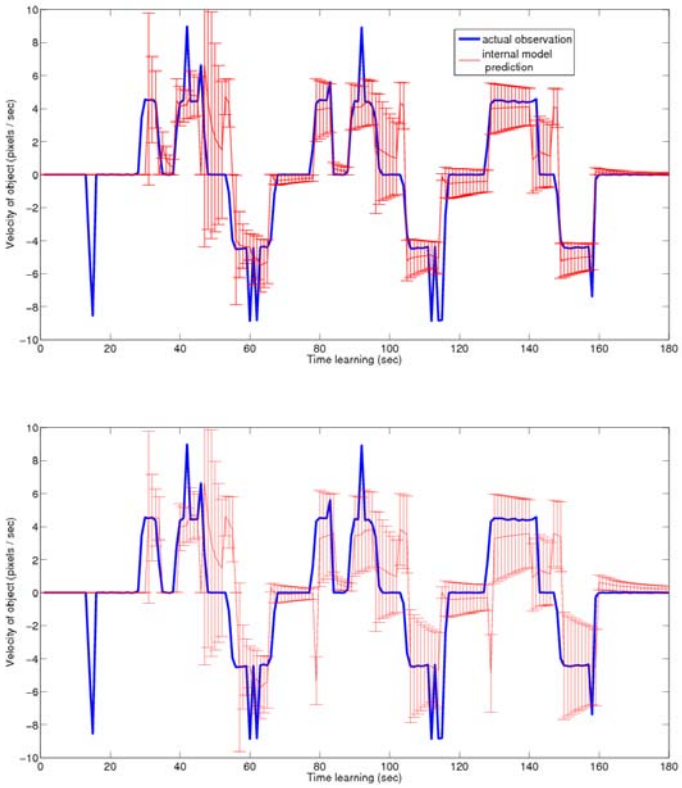


Figure 7. The predictions of two internal model structures for estimating the effect of the velocity motor command as the robot 'babbles' forwards and backwards. The top one can be seen to be the most accurate because it has the lowest estimated prediction variance, shown with the error bars. The structure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

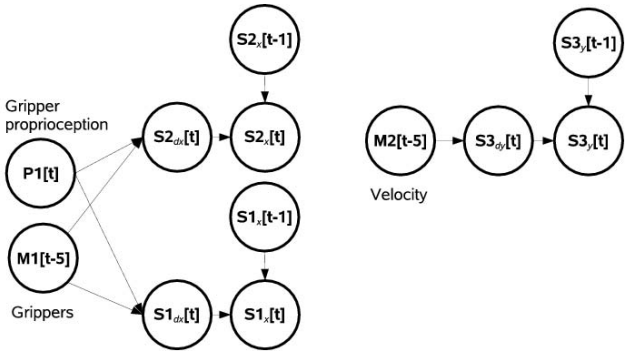


Figure 8. The most useful Bayesian network structures learnt for the gripper motor control (left) and the wheel velocity motor command (right). Both show that the motor commands affect the position of objects in the scene by changing their velocity. It has also learnt that the grippers' touch sensor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

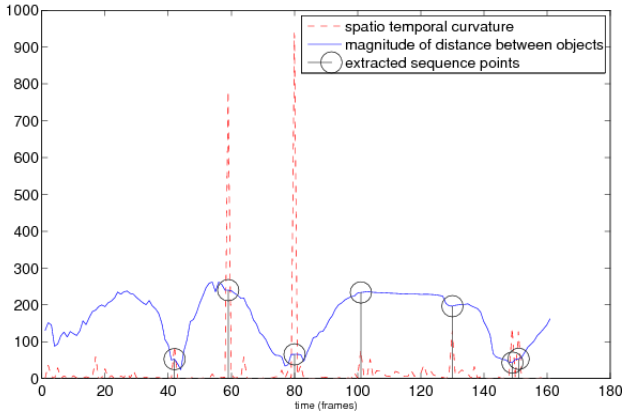


Figure 10. Extracting key points to imitate from an interaction sequence, shown in black circles. These points are extracted from peaks in the spatio-temporal curvature of the distance between the robot's motor system and the object it wishes to interact with.

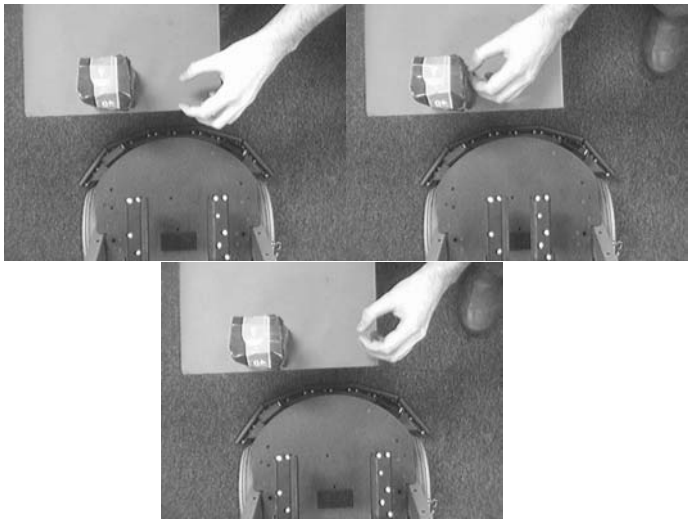


Figure 11. The first three sub-goals being imitated, extracted using the spatio-temporal curvature. Even though this action is occurring as the robot learns to control its gripper system, it is able to recognise it as an interesting action to imitate because neither the human hand nor the pack of biscuits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

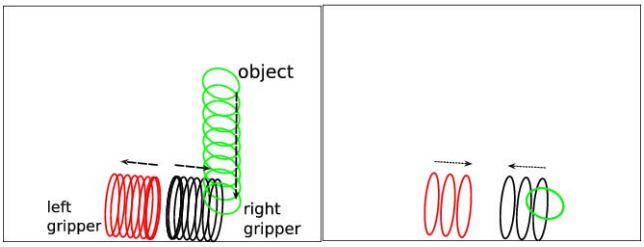


Figure 12. The simulated visualisation of the IOMs as the robot tries to touch the biscuits (left) and then moves away (right). The ellipses represent the means and covariances of the predicted objects' position, and the arrows show the direction of movement. Note that all aspects of this simulation, 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.

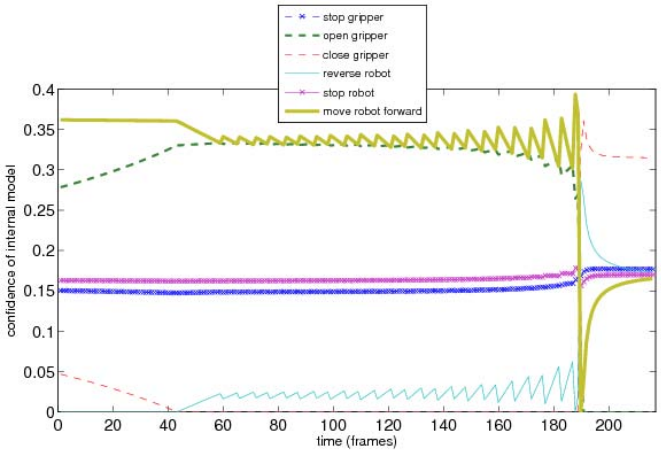


Figure 13. The progress of confidences of each learnt internal model in simulation as the robot tries to touch the object of interest. After this, the first goal state has been reached so the robot moves its grippers away to approach the next sub-goal.

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

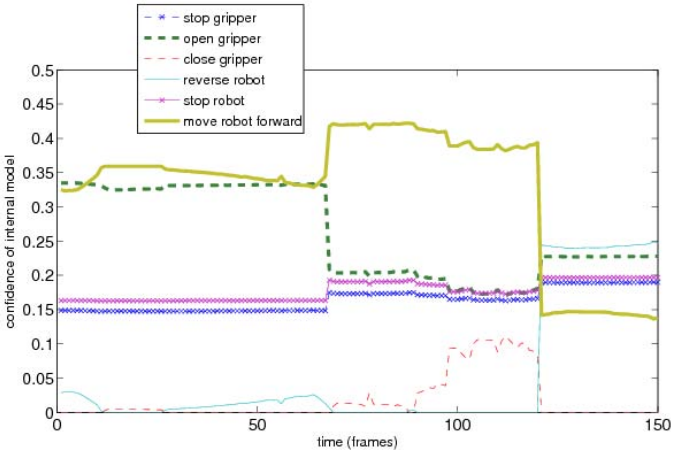


Figure 14. The progress of confidences of each motor command on the actual robot as the robot tries to imitate an interaction with the object.

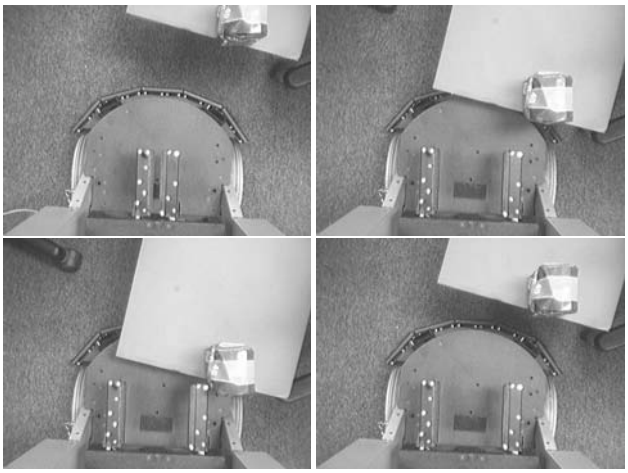


Figure 15. Frames 0, 50, 120 and 150 from the same imitation experiment as as figure 14.

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.

Learning models of camera control for imitation in football matches

Anthony Dearden and **Yiannis Demiris**[1] and **Oliver Grau**[2]

Abstract. In this paper, we present ongoing work towards a system capable of learning from and imitating the movement of a trained cameraman and his director covering a football match. Useful features such as the pitch and the movement of players in the scene are detected using various computer vision techniques. In simulation, a robotic camera trains its own internal model for how it can affect these features. The movement of a real cameraman in an actual football game can be imitated by using this internal model.

1 Introduction

Imitation is a useful way to indirectly transfer knowledge from one agent to another by simply demonstrating the action to the imitator. In this paper, we investigate a particular scenario where this transfer of knowledge can be used to teach robotic cameras how to move in a football match. This scenario has useful applications in both simulation and real-world scenarios. In football computer games such as Pro Evolution Soccer, the movement of the camera during play and automated highlights is generated using pre-programmed control. The movement would be much more natural if it was imitating the movement of actual cameras during a football match. This would also save the programmer the effort of having to create the control algorithms. In actual football matches, up to 20 cameras can be used to provide coverage for a match, each requiring a human operator. Using robotic cameras, automated human-like camera control would give the broadcaster the ability to cover more matches or use more cameras viewpoints. Imitating not just camera movement, but also how the camera shots are selected by a director would enable the entire coverage process to be automated.

It would be advantageous if a robotic camera could be rapidly placed in a viewpoint, learn the effects that it has on a current location and then move accordingly based on the state of the game. To test the feasibility of this approach we implement a learning system on a football simulator, which learns to imitate the camera movements of a trained cameraman, by inverting the learnt effects that its own actions have on the visual field. We test the system on real data, provided by BBC Research, to demonstrate successful learning of the first step of the final system. Current work focuses on understanding the actions of groups of players, so the robotic camera can learn a model of the movement of human cameraman in terms of how the players are moving.

[1] Department of Electrical and Electronic Engineering
BioART group, Imperial College London
E-mail: {anthony.dearden99, y.demiris}@imperial.ac.uk
[2] BBC Research,
Kingswood Warren, Tadworth, Surrey, KT20 6NP
oliver.grau@rd.bbc.co.uk

2 Extracting feature information from real and simulated football games

Before a robotic camera can understand the scene it is in, or is trying to imitate, it is first necessary to extract features from a scene which give the robot information about its own state and the state of other important features like the players. This section describes the image processing steps necessary to extract information about the movement of the camera and the position of players in the game. The same algorithms are applied to both the real and the simulated football match. The real data is taken from the feed from the central 'spotter' camera during the Everton vs Manchester City game in November 2005. The simulation was created using OpenGL to render a basic football game with players and line markings. The list of features which can be extracted from video sequences are shown in table 1.

Table 1. List of features that can be extracted from football video data and the information it provides.

Feature	Information provided
Pitch region information	Position of camera relative to pitch, change of camera shot by director
Skin region information	Close-up shots, crowd shots
Optical flow information	Approximate movement of camera
Player tracking	High-level state of game (e.g. who has possession)

2.1 Finding the pitch and player regions in a video

Figure 1 gives an overview of the computer vision process used to extract player regions. The basic idea behind the process is to subtract the distinctive green colour of the pitch to leave regions which are likely to be players. This idea has been used on numerous previous occasions e.g. [16].

The colour of the pitch is represented as a one- or two-dimensional histogram in HSV colour space. This histogram is back-projected onto each image and then, with a threshold applied, a binary pitch mask can be obtained. To estimate the entire pitch region, the pixels on the binary image are grouped into regions. By calculating the convex hull of the largest regions, the area in the image which the pitch covers can be calculated. Knowing the pitch regions in the image enables the tracking to be simplified by removing clutter from the crowd regions. The shape and position of the pitch region can also give information about the location on the pitch on which the camera is focused. As the colour of the pitch may drift over the duration of the match the histogram can be recursively updated by calculating the histogram of the pitch region excluding player regions.

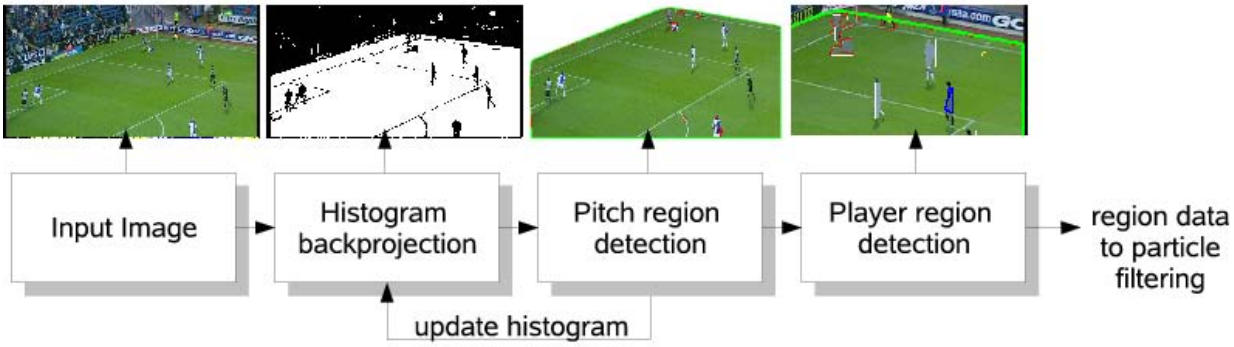


Figure 1. *Overview of the region extraction process*

Once the pitch region has been detected, player regions can be located by finding regions within the pitch region that do not correspond to the pitch colour. The regions can be filtered according to their area in the image, being separated into player regions and 'other' regions. These 'other' regions include noise and regions which are markings on the pitch.

The same technique for extracting the pitch regions can be used to detect regions of skin colour - this is a useful feature for detecting when a particular camera is doing a close-up shot on one of the players.

2.2 Tracking players

Many of the cameras being used to provide coverage for a football match have the sole purpose of tracking the action occurring in the game. Important information about the state of the game can be found from the position and movement of the players on each team; this is obviously an extremely useful feature for any robotic camera wishing to perform imitation.

Tracking footballers in video is made difficult by occlusions; other players or even the referee can obscure the information about a tracked player, as shown in figure 2; this is especially common during tackles, set-pieces and action in front of the goal. Overcoming the problem of occlusions can be done by fusing data from multiple camera sources, with the idea that the ambiguity will not be present from all angles [10, 9]. However, this adds to the complexity of the system; the goal here is to have a tracking system that can work directly from the image from a single moving camera tracking the action. Several techniques have been used previously to disambiguate player regions from a single camera source [7]. The first, also used here, is to apply morphological operators to erode close regions, hoping that they will split apart. Another method is to track the players using a graph representation, whereby the spatial relationship of players before a collision is stored so tracking can be continued when there is no longer an occlusion.

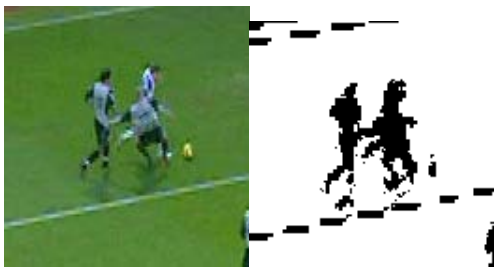


Figure 2. When players occlude each other, maintaining tracking can be difficult as the player region data (right) is ambiguous

To track players, here we use a particle filter. Particle filters have become extremely popular in recent years as a method for Bayesian tracking without the restrictive assumptions of linearity and Gaussian distributions of a Kalman filter [17, 1]. One aspect of particle filters which makes them especially useful in this situation is their ability to simultaneously maintain multiple hypotheses of the state of a tracked object. More details of the algorithm implementation and results can be found in [5].

Figure 3 shows sample frames from the sequences, together with the tracked positions of the two players. The particle filter is able to maintain tracking of both players, despite the occlusion occurring. As expected, when the occluding players separate again, the particles spread into multiple groups because of the increased uncertainty.

2.3 Estimating camera movement in a video

A useful source of information about the position of the camera in the scene comes from how it moves. To extract this information from a video sequence we use the KLT optical flow algorithm to track the movement of pixel-level features in the scene [12]. The pitch and player regions extracted above can be used to limit the points tracked to ones on the pitch; players will usually move independently of the camera. As the real camera moves across the scene, the low-level features leave the field of view, and new, untracked regions enter the scene. The algorithm continuously scans for new features to track and adds them to the list of points being tracked so that there is a continuous stream of tracked point features available. covering the entire image. The information from multiple points can also be combined by taking the average velocity of all points to give an overall metric of the camera's movement.

3 Imitating camera movement

Internal models are structures or processes that replicate the behaviour of an external process [11]. They have been hypothesised to

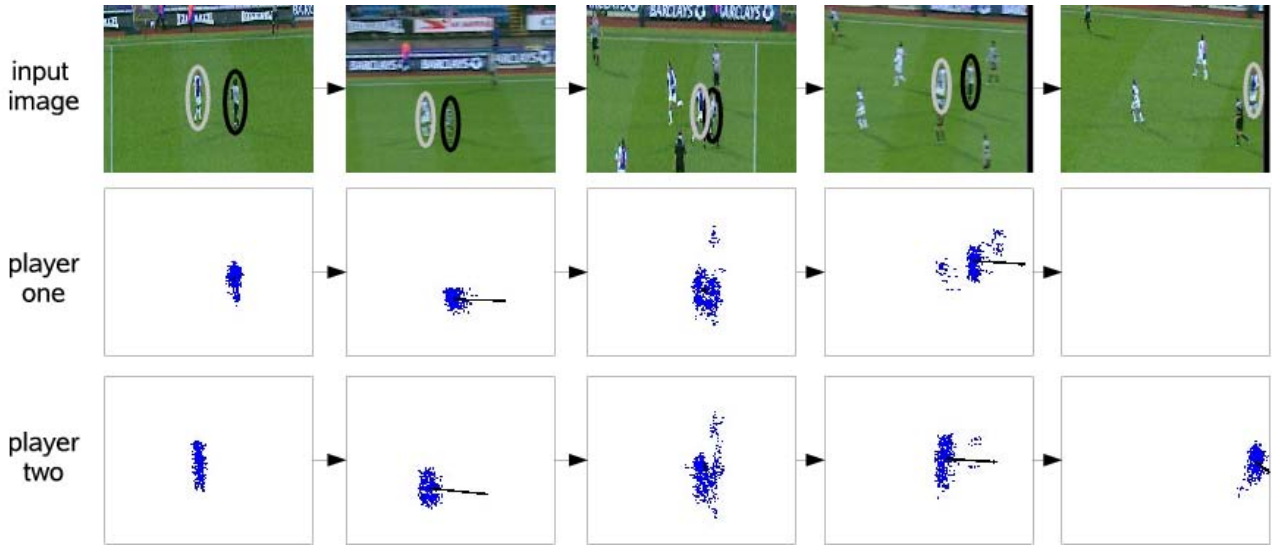


Figure 3. Frames from the tracking of two players. The last frame of player one is empty because the player has left the field of view. The black arrow in the particle represents the estimated velocity of the player. The players being tracked have been manually highlighted in the top images in black for player 1 and grey for player 2

exist in the the human central nervous system, for example to overcome the time delay of feedback from proprioception [18]. Giving a robot the ability to perform an internal simulation of external process enables it to simulate the effects of its actions internally before physically executing them. They enable a robot to predict the sensory consequences of its motor actions as *forward models,* or to estimate the motor commands that will lead to a desired state as *inverse models* [3]. They can be used for imitation by using a *simulation theory* approach [8, 13]. By using the internal models of its own motor system, a robot can understand and therefore imitate the actions it observes a demonstrator taking [6].

An inverse model could be programmed in using hard-coded software to track features on the pitch and thus estimate the position of the camera. The tracking problem in football is quite constrained, and unlike camera movement in other situations, there are reference points in the form of the pitch markings available that could be used. This approach is taken in [15]. A more generic solution would be to allow the robot to learn the internal model for itself through exploration. This would make the system applicable to other situations, and no effort is required by the programmer to come up with an algorithm for the inverse model.

In this work, the robot's actions are its pan, tilt and zoom commands; the camera is assumed to be stationary in the scene; a valid assumption for most cameras used in a football match. The sensory information it receives is provided by the computer vision features described in section 2, and listed in table 1. In this initial work we will just be focusing on using the optical flow information. The robotic camera needs to learn internal models which represent the effects its motor commands have on the optical flow data it receives back.

The internal models are represented either with radial basis functions or using the non parametric K-Nearest Neighbour (KNN) algorithm [2]. Radial basis functions had the benefit of being naturally smooth function approximators, whereas the KNN algorithm trains much faster[3], and allows the learnt forward model to be easily inverted and used as an inverse model to predict the motor command that can be used to recreate a particular movement. The KNN algorithm was implemented by storing the set of previous motor commands, the *pan* and tilt values, and the corresponding feature vectors, the optical flow velocity vector of image. To use a set as a forward model is a case of finding the K motor commands nearest to the one to be predicted for, each having a distance, d from the desired command. The corresponding K features for each of these commands can then be averaged to provide the predicted feature outputs. The average was weighted using a Gaussian kernel according to the distance, d. To use the KNN technique for an inverse model requires performing the process in reverse.

To train the internal model, the robot needs to execute multiple motor commands to produce a corresponding set of sensor data. In previous work [4], exploration of the motor-space with a camera was performed optimally so as to minimise the error in the internal model. As the only results currently available are on a simulated camera, the time taken for each camera to learn the internal model is less critical. Furthermore, only 2 degrees of motor freedom were involved. Therefore random motor commands were used to provide the training data.

The robotic camera uses the internal model it has learnt to imitate the movement of a trained camera man, and the optical flow features from the movement of the real camera man are given to the inverse model of the robotic camera. This will then output the motor commands the model expects will most likely recreate this movement in the robotic camera. The overview of this process is shown in figure 4. Selected screenshots for the simulated robotic camera imitating a real camera man are shown in figure 5. The left images are taken from the movement of the professional cameraman, and the right images show the simulated robotic camera's attempt to imitate the movement. Using only the optical flow features for imitation has the benefit of the robotic camera producing smooth, human like movement. Work is currently ongoing to make use of other features to ensure that absolute position information is used; as can be seen by the last frame, the imitating camera has drifted significantly from the camera it is imitating.

[3] training speed on a simulated camera is less of an issue than on an actual robot, where training time is limited

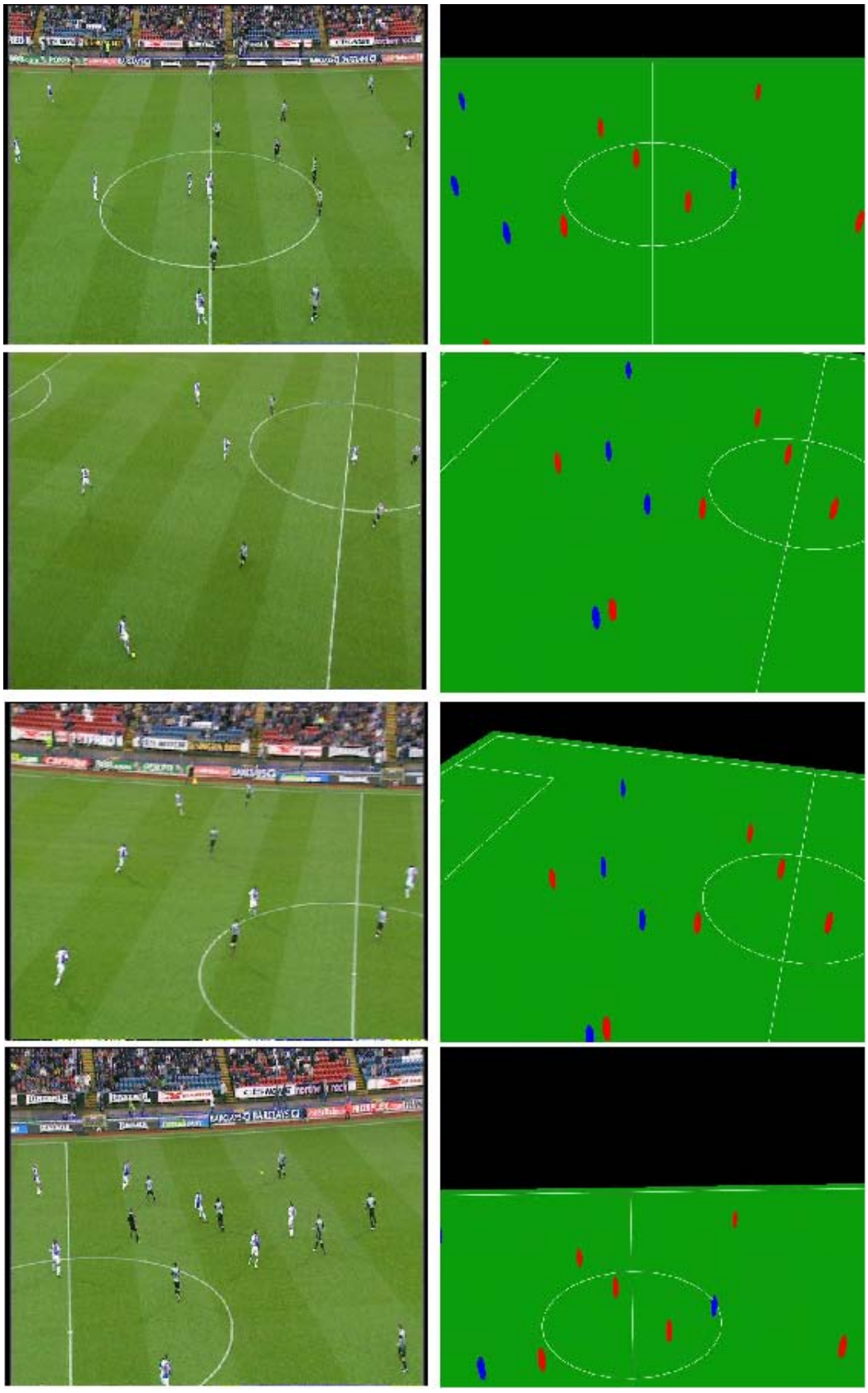


Figure 5. Frames 0, 100, 200 and 300 from the real football match and the imitating camera in simulation. The movement of the robotic camera is quite smooth and 'human-like'. However, as the movement is imitated using dynamic information, the absolute error in the robotic camera's position begins to accumulate.

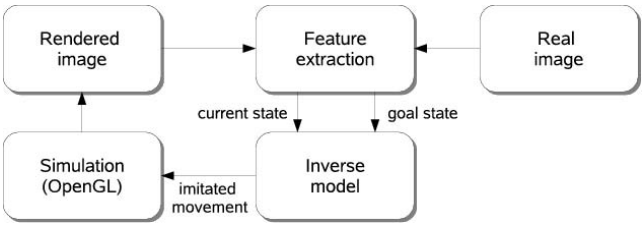


Figure 4. The imitation process using the learnt inverse model.

4 Discussion

The imitation the system performs so far is a mapping of one camera movement onto another. For imitation to be more general, the internal models need to be learnt at a level of abstraction at which they are applicable to any particular football match. It is intended that the robotic camera would be capable of tracking the action in a football match based on the actions taken by a professional cameraman with respect to the current state of the game. Much of the work on extracting information on the state of the game has been completed; the position of players on the pitch provides the most useful information for this. Work is currently ongoing to augment the structure of the inverse model so that camera movement is learnt as a function of player movement, I.E, given how players are currently moving, a robotic camera can move in the same way a human would move in the same situation.

Beyond the level of the movement of an individual camera, there is also the issue of how a human director switches between and sends requests to each camera. We are working to produce a system that can model and imitate this. The feature data that can currently be extracted provides useful information about the actions of the director. Figure 6, for example, shows how one of the features, the amount of

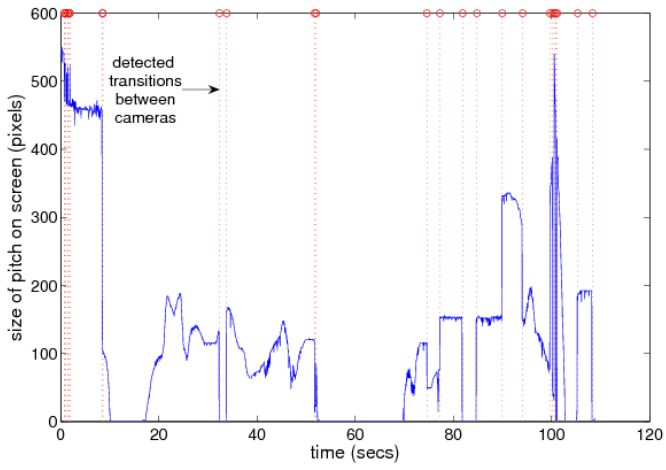


Figure 6. How the size of the pitch detected on screen varies over time. Rapid changes in this value can be used to detect when the director has switched between cameras

pitch visible in the broadcast footage, varies over time. By detecting rapid changes in this value, it is easy to split the final footage into individual camera shots. A promising method for modelling these scene changes at a higher level is the use of dynamic Bayesian networks, such as hidden Markov models [14]. The switching between cameras given the state of the game can be modelled as sequence of hidden discrete states. The transition model for these states - i.e. how a human director switches between shots can be learnt using the low level features described in this work as the training data.

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 for their comments, and BBC Research and Match of the Day for access to the video footage.

References

[1] M. S. Arulampalam, S. Maskell, N. Gorden, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE transactions on signal processing*, 50(2), 2 2002.

[2] C. G Atkeson and S. Schaal. Memory-based neural networks for robot learning. *Neurocomputing*, (9):1–27.

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

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

[5] 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.

[6] 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.

[7] P. Figueroa, N. Leite, R. M. L. Barros, I. Cohen, and G. Medioni. Tracking soccer players using the graph representation. In *Proceedings of the 17th international conference on pattern recognition (ICPR04)*. Los Alamitos, Calif.; IEEE Computer Society, 2004.

[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] S. Iwase and H. Saito. Tracking soccer player using multiple views. In *IAPR workshop on machine vision applications*, 2002.

[10] S. Iwase and H. Saito. Tracking soccer players based on homography among multiple views. In *Visual Communications and Image Processing*, pages 283–293, 2003.

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

[14] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 2 1989.

[15] G.A. Thomas. Real-time camera pose estimation for augmenting sports scens. In *Proceedings of the 3rd European Conference on Visual Media Production (C*, pages 10–19, 2006.

[16] O. Utsumi, K. Miura, I. Ide, S. Sakai, and H. Tanaka. An object detection method for describing soccer games from video. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*. IEEE, 2002.

[17] J. Vermaak, A. Doucet, and P. Perez. Maintaining multi-modality through mixture tracking. In *Ninth IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2003.

[18] D. M. Wolpert, R. C. Miall, and M. Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, September 1998.

Imitating the Groove: Making Drum Machines more Human

Axel Tidemann[1] and **Yiannis Demiris** [2]

Abstract. Current music production software allows rapid programming of drum patterns, but programmed patterns often lack the *groove* that a human drummer will provide, both in terms of being rhythmically too rigid and having no variation for longer periods of time. We have implemented an artificial software drummer that learns drum patterns by extracting user specific variations played by a human drummer. The artificial drummer then builds up a library of patterns it can use in different musical contexts. The artificial drummer models the groove and the variations of the human drummer, enhancing the realism of the produced patterns.

1 Introduction

Our motivation for creating an artificial drummer was to combine the low-cost approach of programming drum parts through Digital Audio Workstations (DAWs, such as Pro Tools[3], Logic[4], Cubase[5], Digital Performer[6]) with the groove that a human drummer will provide. When producing music, recording the drums is a time-consuming and expensive process. The drums must be set up in a room with suitable acoustics and high quality microphones in order to produce good sounding drums. Subsequently, the drummer must play the actual part that is to be recorded. Most drummers do not play an entire song without any flaws, so the actual recording is also time-consuming. The current DAWs allow for cut-and-paste editing of the recorded audio, so a perfect take of a song is not required to produce a good result. This has drastically reduced the time required to record music in general, not only drums. But still the cost of recording drums is high, so for producers it is often more desirable to *program* the drums in the DAW. This approach is very low-cost, but it is often difficult to get a result similar to that of a real drummer. Programmed patterns have perfect timing and the velocity (i.e. how hard a note is played) of the beats is the same. A human drummer will always have small variations in both timing and velocity of each beat, which is often described as the *feel* or *groove* of the drummer. In addition, a human drummer will vary what he/she plays, such as adding an extra snare drum[7] beat or a fill when playing a certain pattern.

Programmed patterns can be altered to mimic these variations, but this requires the producer to manually change the velocity and timing of each beat, in addition to adding or removing beats to create variations. This can be very time-consuming, and requires musical knowledge of how to produce variations that will be perceived as those of a human drummer. Current DAWs have the ability to alter the beats by adding random noise, which might provide a more human-like feel to the drum tracks since the added noise will be perceived as human flaws. However, there is no guarantee that the result will sound more human-like, since the DAW itself has no understanding of what makes a drum pattern sound like it was played by a human. The research goal of this paper is to make an artificial drummer that is able to play patterns with feel and variation. This is realized by making the artificial drummer learn drum patterns from human drummers. The artificial drummer will *model* the variations that provide the feel of the drum pattern, which it can use to imitate the drumming style of the human drummer.

2 Background

The music software industry has created more complex samplers and synthesizers over the years as computers have become an important tool for musicians. To recreate the sound of a drumkit, a lot of effort has gone into recording huge libraries with gigabytes of samples (e.g. FXpansion BFD[8], Toontrack dfh[9], Reason Drum Kits[10], Native Instruments Battery[11]). The samples are then *layered* to simulate the dynamics experienced when playing real drums, i.e. that the pitch changes when playing soft or hard. Typically, when playing the snare drum in one of the aforementioned libraries, it will consist of a multitude of samples to achieve a more life-like response to playing dynamics.

These libraries are very sophisticated and sampled with meticulous precision, but they still need to be programmed. Even though these libraries come with software interfaces that are easy to program (most of them even come with rhythm pattern templates), there is still no substitution for a *real* drummer: the libraries themselves are merely tools for reproducing drum sounds, and the software interfaces have no intelligent way of generating human-like drum patterns. The templates will often be too rigorous and lifeless, something patterns programmed by the user also often suffer from (unless the user manually changes every note in the patterns generated, a very time-consuming process).

If the groove of a drummer could be *modeled*, a studio producer would have access to an artificial drummer that would be more life-like than what is currently available. The artificial drummer would

[1] SOS Group, IDI, Norwegian University of Science and Technology. Email: tidemann@idi.ntnu.no
[2] BioART, ISN Group, Department of Electrical and Electronic Engineering, Imperial College London. Email: y.demiris@imperial.ac.uk
[3] http://www.digidesign.com
[4] http://www.apple.com/logicpro/
[5] http://www.steinberg.net/
[6] http://www.motu.com/
[7] A drumkit typically consists of a kick drum (which produces the low-frequency "thud" sound), a snare drum (a more high-pitched crackling sound) and a hihat (a high-frequent "tick" sound), see figure 3.
[8] http://www.fxpansion.com/index.php?page=30
[9] http://www.toontrack.com/superior.shtml
[10] http://www.propellerheads.se/products/refills/rdk/index.cfm?fuseaction=mainframe
[11] http://www.native-instruments.com/index.php?id=battery_us

be able to imitate a certain style of playing, specific to the drummer it has learned from. For producers, this would lower the cost of having life-like drums, and the producer could have the drummer of his choice to perform with the drummer's unique style. A professional drummer will have the opportunity to teach the artificial drummer his own unique style of playing, which he/she could later use in the studio or sell as a software plug-in.

We will now present a brief overview of research done in modeling the expressive performance of musicians. Saunders et al. [17] use string kernels to identify the playing style of pianists. The playing style is identified by looking at changes in beat-level tempo and beat-level loudness. However, imitating the style of the pianists was not attempted. Tobudic and Widmer also consider variations in tempo and dynamics as the two most important parameters of expressiveness. To learn the playing style of a pianist, they use first-order logic to describe how the pianist would play a certain classical piece, and then a clustering algorithm to group similar phrases together [19, 18]. They use the models to play back music in the style of given pianists, but some errors arise during playback. Tobudic and Widmer admit that these errors are due to the modeling approach (in fact, in [19] they claim it is "not feasible" to model the playing style of a pianist with the current data and training methods; the modeling approach was deemed too crude by the authors to be used as sufficiently accurate training data). Pachet's Continuator uses Markov models to create a system that allows real-time interactions with musicians [3, 5, 2], however his focus is more on replicating the tonal signature of a musician; the Markov model represents the probabilities that a certain note will follow another. A musician plays a phrase (i.e. a melody line), and the Continuator will then play another phrase which is a *continuation* of the phrase played by the musician (hence its name). Mantaras and Arcos use case-based-reasoning to generate expressive music performance by imitating certain expressive styles, such as joyful or sad [16, 15, 13, 12].

As far as the authors know, modeling the style of drummers is a novel approach to create an artificial drummer. The Haile drummer of Weinberg [23, 22] has some similarities, but there are some major points that separate it from our approach: first of all, it is a percussionist. Haile is a robot that plays a Native American Pow-wow drum, it uses only one arm and is far from being full-fledged drummer. In addition, it does not learn its patterns from human input, it has a database of rhythm patterns that are constructed by the designers of the system. Haile does imitate and modify patterns when interacting with human players, but it does not *learn* these patterns.

3 Architecture

We call our architecture "Software for Hierarchical Extraction and Imitation of drum patterns in a Learning Agent" (SHEILA). The following section will explain this architecture in more detail.

3.1 Input

Drum patterns are given as input to SHEILA. Ideally, the drum patterns would be extracted from audio files, however in this paper we have used MIDI[12] files as input to SHEILA. MIDI is a symbolic representation of musical information, and since it incorporates both timing and velocity information for each note played, it is very well suited for this application. SHEILA processes the MIDI file and learns the style of the human drummer.

[12] Musical Instrument Digital Interface, a standard developed in the 1980s to enable communication between electronic music equipment.

Another advantage with representing the drum patterns using MIDI is that it is a tempo-less representation. Once SHEILA has learnt a pattern, it can be played back at a different tempo then when it was demonstrated, which gives the producer even greater flexibility.

3.2 Modeling

The system operates at two levels by modeling small and large scale variations, which will now be explained.

3.2.1 Small-scale variations

The *small-scale variations* arises as follows: when a drummer plays a specific pattern, he/she will play each beat of the pattern slightly different each time. The differences will occur in both timing and velocity. By calculating the mean and standard deviation of both the velocity and timing of each beat over similar patterns, the small-scale variations can be modeled using the Gaussian distribution. We investigated whether the Gaussian distribution was an appropriate model for the data by playing quarter-notes for about 8 minutes at 136 beats per minute (BPM), yielding 1109 samples. The histogram of the onset time and the velocity can be seen in figures 1 and 2 respectively, showing that the normal distribution is an appropriate model of the data.

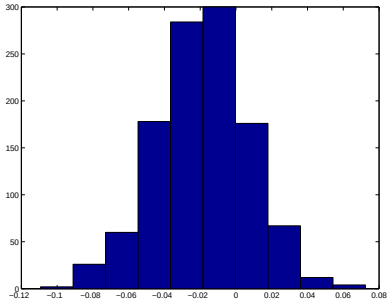


Figure 1. The histogram of the onset time after playing quarter notes for 8 minutes. The bars show distribution of the timing of the beats relative to the metronome.

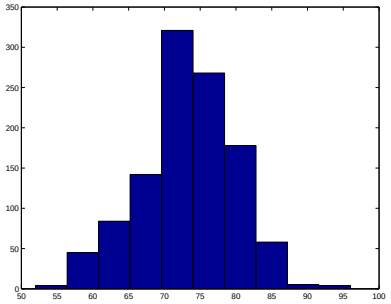


Figure 2. The histogram of the velocity after playing quarter notes for 8 minutes. Note that the histogram is not as nicely shaped as that of the onset time. This is most likely due to the velocity sensitivity in the pads that were used for gathering MIDI data, something that does not affect the onset time for each beat. The pads of the Roland SPD-S (see section 4 for description of the equipment) used in the experiment are rather small, and hitting towards the edge of the pad will affect the recorded velocity, even though the drummer might have hit the pad equally hard each time. Still, the histogram clearly shows the Gaussian bell-shaped curve for the samples gathered.

3.2.2 *Large-scale variations*

Variations of the pattern itself, i.e. adding or removing beats are considered to be *large-scale variations*. Variations of a pattern is then stored along with the pattern it is a variation of, and based on a calculated probability, SHEILA will play back a variation of a certain pattern instead of the pattern itself. Exactly how this is done is elaborated on in the next section.

3.3 Training

To train SHEILA, the drum track of a song is given as input. In pop and rock music it is very common to divide a song into parts, such as a verse, chorus and a bridge. The song used in the experiments (see section 4) has the following structure: verse/chorus/verse/chorus/bridge, which is a common structure in pop and rock music. The point is that the drummer plays different patterns for the verse, chorus and bridge. We will now explain how SHEILA learns both large-scale variations of patterns and the small-scale variations of each pattern.

3.3.1 *Learning large-scale variations*

The occurrence of each of the patterns in the song is calculated (one pattern is then defined to be one measure, i.e. 4 quarter notes long). The patterns that are most frequently played are then considered to be *core patterns*. For instance, in a certain song the first core pattern C_1 occurs at measure 1. If the next core pattern C_2 appears at the 8th measure, the patterns that differ from C_1 between measure 1 and 8 are considered to be *large-scale variations* of C_1, named C_1V_x, where x is increasing with the number of variations of C_1. The ratio of variations of the core pattern (r_v) is calculated. This ratio will indicate how often a core pattern is to be varied when SHEILA will imitate the core pattern.

3.3.2 *Learning small-scale variations*

For each of the patterns (i.e. both core patterns and their variations), the mean (μ) and standard deviation (σ) of both the onset time and velocity is calculated, representing the *small-scale variations*. This is calculated the following way: the similar patterns are grouped together, and for each beat in the pattern, the mean and standard deviation for both velocity and onset time is calculated across the similar patterns. In order to calculate the mean and standard deviation of the onset time, a copy of all the patterns is quantized. Quantization means shifting each beat to the closest "correct" beat. If a beat was supposed to be on the "1", and it was slightly before or after, it is shifted to be exactly on the "1". The difference between the quantized pattern and the actual pattern is used to calculate the mean and standard deviation of the onset time for each beat. Each pattern (be it core or variation) will then have the normal distribution parameters assigned to each beat. An "ideal" (i.e. quantized and with no velocity information) version of this pattern is then stored in the SHEILA library, along with the mean and standard deviation of both onset time and velocity for each beat. A simplified outline of this procedure can be seen in algorithm 1. When imitating this pattern, the assigned parameters of the normal distribution will then be used to shift the beat forwards and backwards in time and to calculate the velocity. This will be explained further section 3.4.

3.3.3 *Creating a library of the patterns*

After processing the MIDI file, SHEILA will have built up a library of core patterns and their variations, see figure 3. SHEILA also stores which core patterns make up a song. This is simply an aid for the user of SHEILA; if the user knows the song the drum pattern was learned from, he will instantly know what kind of style the pattern was played in. In addition, SHEILA stores the name of the drummer playing this pattern. This is because it is very likely that different drummers will play the same pattern. SHEILA will model how each of them played the same pattern, and the name of the drummer can be presented to the user of SHEILA to further aid the user in indicating what kind of style the imitated drum patterns will be in.

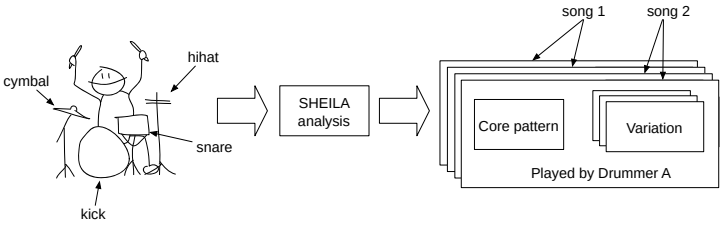


Figure 3. The learning process. Drum patterns are input to SHEILA, which analyzes the patterns and stores them in a library.

Algorithm 1 Training

```
count occurrence of each pattern in song
core patterns = most frequently played patterns
collect core patterns and their variations in groups
for all groups do
    calculate μ, σ of onset time and velocity for each beat across patterns (i.e. small-scale variations)
    store core pattern and variations (i.e. large-scale variations) along with μ, σ of each beat in SHEILA
end for
```

3.4 Imitation

This section describes how SHEILA can be used to imitate a given drum pattern in the style of a specific drummer.

3.4.1 *Selection of playing style*

If a producer wants SHEILA to play a certain pattern, he can write it down in a sequencer, export the pattern as a MIDI file and give it to SHEILA. If the pattern is recognized in the SHEILA library, it can then imitate the pattern in the style of the drummer that served as a teacher for the pattern. Indeed, if SHEILA recognized several drummers that played the same pattern, the producer will have the choice of selecting between the different drummers. The name of the song is also stored along with the drum patterns, allowing the producer to quickly have an idea of what the resulting pattern would sound like (presuming the producer knows the song). A good example is the pattern shown in figure 6. For many drummers, this is the first pattern learnt, and it is widely used in pop and rock music. If SHEILA had learnt the styles of all the major drummers in recorded music history, it would give the producer the choice of generating this pattern as played by Ringo Starr on "Help!" (the drummer of The Beatles,

i.e. sloppy timing and simple variations) or Lars Ulrich on "Sad But True" (the drummer of Metallica, i.e. a rather "heavy" groove that is slightly behind the time, with typical heavy metal variations), among others. This is shown to the left in figure 4.

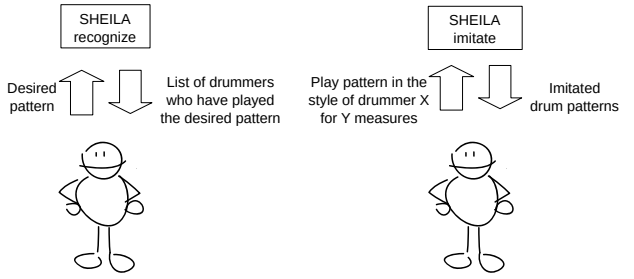


Figure 4. Two steps that allows SHEILA to imitate a certain drummer. To the left the producer decides he wants SHEILA to play a specific pattern. He inputs this pattern in the MIDI format to SHEILA, which recognizes the pattern. Often several drummers will have played this pattern, and the output is a list of the drummers who can play this pattern and in which song it appeared. To the right shows the producer deciding which drummer should be imitated when generating the patterns, and he inputs this along with how many measures the pattern should be played for. SHEILA then imitates the style of the drummer specified, and outputs the imitated drum patterns back to the producer, ready to be used in the DAW of his choice.

3.4.2 Generation of patterns

Once the producer has decided which of the drummers in the SHEILA library he wants to use, he tells SHEILA to play the desired pattern in the style of drummer X for Y measures. At each measure, SHEILA decides whether to play the core pattern or one of the variations of the core pattern. The ratio of variations of a core pattern serves as the probability that a variation of the core pattern is played instead of the core pattern. The next step is to generate the actual beats that make up a pattern. When a pattern is to be generated, the onset time and velocity of each beat are calculated by generating random numbers from a Gaussian distribution, using the mean and standard deviation stored for each beat as parameters. This will yield slightly different patterns each time they are generated, but they will still sound similar, since the generation of patterns will come from a model of how the human drummer would play it. See algorithm 2 for a simplified description. The generated drum patterns are written to a MIDI file, which can later be imported into a DAW with high quality drum samples.

Algorithm 2 Imitation

```
present pattern p to be imitated to SHEILA
if p is known then
    make user select which drummer should be used for imitation
    of p, and for how many bars
    for the desired number of bars do
        if random number < r_v then
            generate variation of p using the stored μ, σ
        else
            generate p using the stored μ, σ
        end if
    end for
end if
return generated patterns
```

3.5 Implementation

The SHEILA system was implemented in MatLab, using the MIDI Toolbox [10] to deal with MIDI file input/output. Propellerheads Reason 3.0 was used for recording MIDI signals and for generating sound from MIDI files, as explained in the following section.

4 Experimental setup

To acquire drum patterns, we used a Roland SPD-S which is a velocity sensitive drum pad that sends MIDI signals. Attached to the SPD-S was a Roland KD-8 kick drum trigger, along with a Pearl Eliminator kick drum pedal. A Roland FD-8 was used as a high hat controller. An Edirol UM-2EX MIDI-USB interface was used to connect the SPD-S to an Apple iMac, which ran Propellerheads Reason 3.0 as a sequencer, recording the MIDI signals. Reason was loaded with the Reason Drum Kits sample library to generate sound from the MIDI signals. The drummer would listen to his own playing using AKG K240 headphones connected to the iMac. The setup can be seen in figure 5.

Three drummers were told to play the same song, i.e. the same patterns for the verse, chorus and bridge, yielding three core patterns. If the verse is C_1, the chorus C_2 and the bridge C_3, then the structure of the song looks like this: verse (i.e. C_1) 8 measures, chorus (i.e. C_2) 8 measures, verse 8 measures, chorus 8 measures and finally the bridge (i.e. C_3) the last 8 measures. The drummer played along with a metronome to ensure that the tempo was kept constant. Each drummer would play in the tempo that felt most natural, so the tempo was varied around 100 beats per minute. After playing, the MIDI file was given as input to SHEILA. The pattern for the verse is shown in figure 7.

5 Results

This section is divided in three; the first two show how SHEILA models the drummers and how these models can be used to imitate the playing style of different drummers. The last section demonstrate listeners' ability to recognize which human drummer served as a teacher for the imitated patterns.

5.1 Modeling

Since all drummers played the same pattern, it is possible to see how SHEILA models each drummer differently. Figures 9-11 show the mean and standard deviation of the velocity for each beat when playing the pattern shown in figure 7 for drummers A, B and C respectively. Note that the scale along the Y axis is $[0 - 127]$, which is the range of the MIDI signal. The figures also show the mean and standard deviation of the onset time of each beat. The velocity bar is plotted on the mean onset time, which is why the velocity bars are not exactly on the beat. The standard deviation of the onset time is shown as the horizontal lines plotted at the base of each velocity bar (see figure 8 for a zoomed in plot with descriptive arrows that will help understand the plots). This is most clearly visible for drummer A (figure 9). Figures 12-14 more clearly show the mean and standard deviation of the onset time. The differences from 0 is how much the drummer is ahead or lagging behind the metronome. Between each quarter note beat (i.e. 1, 2, 3, 4) there are 100 ticks, divided in the range $[0 - 0.99]$. Since the data gathered is in the MIDI format, a tick is not a unit of time until the tempo has been decided. We present the results in ticks instead of another unit such as milliseconds, since

Figure 5. Playing drum patterns on the Roland SPD-S.

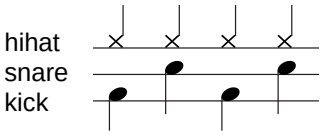


Figure 6. A simple and common drum pattern in pop and rock music.

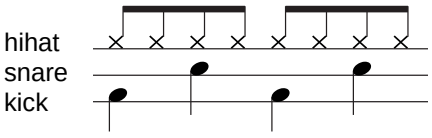


Figure 7. One of the patterns played by all the drummers in the experiments.

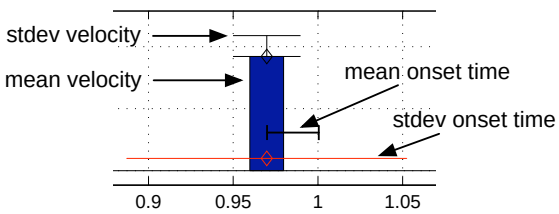


Figure 8. A zoomed in version of the third plot in figure 9. The arrows show how the mean and standard deviation of both the velocity and onset time is plotted. Note that the bar showing the mean onset time is *not* plotted on the figures, this is shown simply as the displacement from the nearest 8th note value (1 in this figure). These displacements are most easily seen in figure 9, for drummer B and C the displacements are smaller and are more easily observable on the onset time plots.

the ticks will accurately show the relative difference between each drummer, regardless of tempo. Drummer B has a mean onset time of -0.034 for the first kick drum beat (figure 13). This may not seem like a big difference, but these small variations are easy to pick up on when listening to a drum pattern. In fact, they are a crucial element to the groove of the pattern. MP3 files are available[13] that better illustrate these differences.

The figures clearly show how each drummer has his unique style. This is most easily seen on the hihat beats, as the accentuation is very different from drummer to drummer. Drummer B has a classic rock style of playing the pattern, with heavy accentuation on the quarter note beats (1, 2, 3, 4) and lighter notes on the off-beats (i.e. the *and* between the quarter notes), see figure 10. Figure 13 shows that he is constantly slightly ahead of time, which adds more aggressiveness to the playing style, and is also very common in rock music. Drummer A (figure 9) has a more even feel and is the drummer that varies most in timing (figure 12). This allows for a more relaxed feel, but will most likely sound rather sloppy when played at a high tempo.

Drummer C has the onset time mean closest to zero of all the drummers, see figure 14. Since he is both slightly ahead and behind the metronome it does not sound as tight as drummer B, which is constantly ahead of the beat. Instead, it has a more laidback feel that sounds more natural when played back at lower tempos.

It must be noted that describing the qualities of each of the drummers is inherently vague, but the graphs show that SHEILA successfully models the different styles of the drummers. Again we refer to the available MP3 samples.

5.2 Imitating

The models acquired for each drummer can now be used to *imitate* them. The imitation will be of both the small-scale variations (i.e. small changes in velocity and onset time in a pattern) and large-scale variations (varying the core pattern). To see how the large-scale variations are introduced, a simple experiment was done. After SHEILA had modeled each drummer playing the same song, SHEILA was used to imitate each drummer playing the same song all over again. Recall from section 4 that the structure of the song was playing verse/chorus/verse/chorus/bridge, each for 8 measures, and that the verse, chorus and bridge corresponded to C_1, C_2 and C_3 respectively. To imitate the same song, SHEILA was then told to play the same song structure (i.e. C_1 for 8 measures, C_2 for 8 measures, C_1 for 8 measures and so on). How the song was originally played along with the large-scale variations introduced when imitating the style for each drummer is shown in table 2.

Figures 15-17 show how the pattern in figure 7 was played back differently in terms of small-scale variations for each of the drummers. The figures show only one measure, over several measures these would be slightly different. They can be compared to figures 9-11, which show the mean and standard deviation of the velocity and onset time. Likewise, the onset time from the imitated pattern is shown in figures 18-20.

5.3 Evaluation by listeners

In order to examine how well SHEILA imitates the playing style of the three different drummers, we got 18 participants to compare the output of SHEILA to that of the original drummers. In order to make it harder to tell the drummers apart, the listeners heard 8 bars

[13] http://www.idi.ntnu.no/∼tidemann/sheila/

of each drummer played at 120BPM, yielding 15 second samples of drumming. The same drumkit sample library was used to create identically sounding drumkits. The drummers originally recorded their drumming at different tempos (e.g. the tempo that felt most natural to them). Since the drumming was recorded in the MIDI format, it could be sped up without any distorted audio artifacts.

SHEILA then generated another 8 bars in the style of each drummer, played back at 120BPM. This included large-scale variations that were not present in the 15 second samples that the listeners would use to judge the imitation by. The evaluation was done as follows: the participants listened to the samples of the original drummers, and then the imitated patterns produced by SHEILA, which were presented in random order. The participants were free to listen to the samples in any order and as many times as they liked. The listeners completed the experiment by classifying each of the imitated drum patterns as being that of drummer A, B or C.

Table 1 shows that the listeners correctly classified which drummer served as a teacher for the imitated drum parts most of the time; the lowest classification rate being that of drummer C which was 83.3%.

Drummer	A	B	C
Classification	94.4%	88.9%	83.3%

Table 1. How often the imitated SHEILA output was correctly classified as being imitated from the corresponding human drummer.

6 Discussion and conclusion

We have implemented an artificial drummer that learns drum patterns from human drummers. In addition to simply learning the drum patterns themselves, the system *models* how a drummer would play a certain pattern, both in terms of small-scale variations in timing and velocity, and large-scale variations in terms of varying patterns. This has been demonstrated by letting three different drummers play the same song, and then showing how SHEILA models the different style of each drummer. Subsequently, we showed how SHEILA will play back the same song in a different way (in terms of large-scale variations), and also how the imitated pattern themselves are slightly different in terms of small-scale variations, but still in the *style* of the imitated drummer. By human evaluation, we have shown that the imitated drum patterns are often perceived as being similar to the originals. The work presented in this paper has demonstrated the core principle for using learning by imitation: namely to simply show the computer what you want it to do, and them make it imitate you.

Note that SHEILA need not be trained only on songs. For instance, to model how a certain drummer would play the pattern shown in figure 7, the drummer could play the pattern for a certain amount of measures, adding the large-scale variations the drummer would feel natural to play with this pattern. This would be a useful approach in terms of building up huge libraries of patterns and variations of these patterns, but this lacks the aspect of how the drummer played in order to fit the musical context. The advantage of training SHEILA based on patterns in a song is that the producer using SHEILA to generate drum patterns will instantly know which feel was on that track, and there would not be variations that will appear out of context.

The MIDI files used in this experiment was made by amateur drummers, since hiring professional drummers would be too expensive. The MIDI representation has the advantage of being tempo-less,

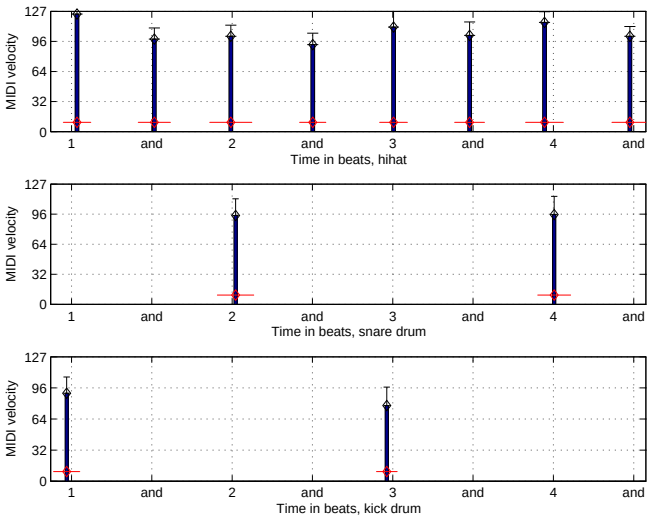


Figure 9. Velocity and onset time plot, drummer A. The hihat velocity is not varied to a great extent, but with more variance in the onset time gives the playing style a relaxed feel. Recall that the Y scale is $[0-127]$, which corresponds to the MIDI resolution. The X scale corresponds to the beats in the measure.

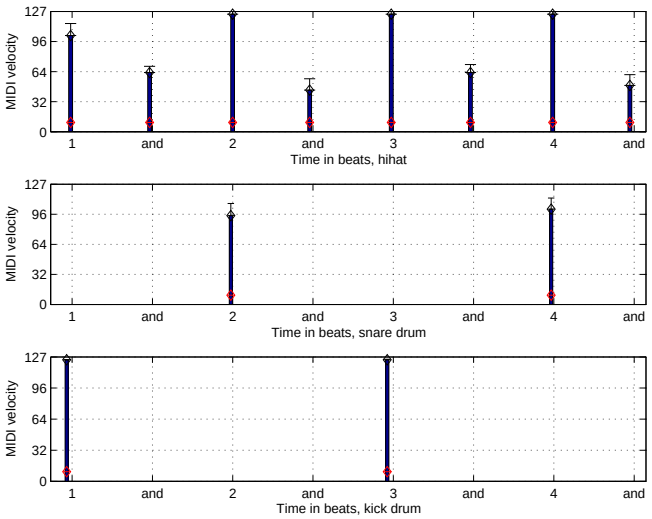


Figure 10. Velocity and onset time plot, drummer B. The hard accentuation on the downbeat is common for rock drummers.

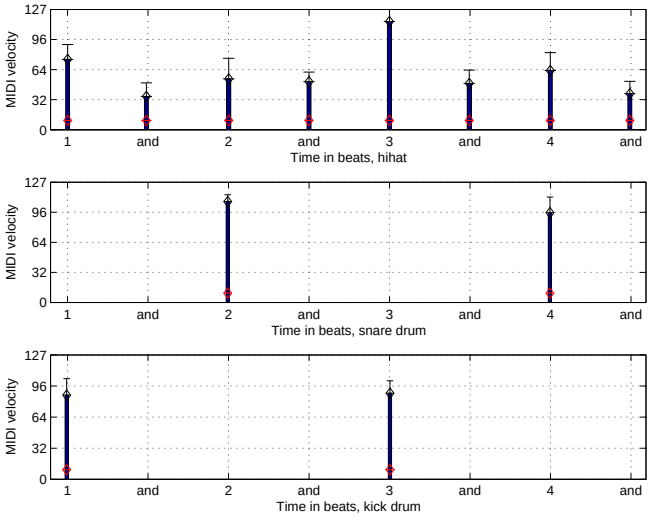


Figure 11. Velocity and onset time plot, drummer C. A more odd variation of velocity for the hihat, which creates a rather laidback feel.

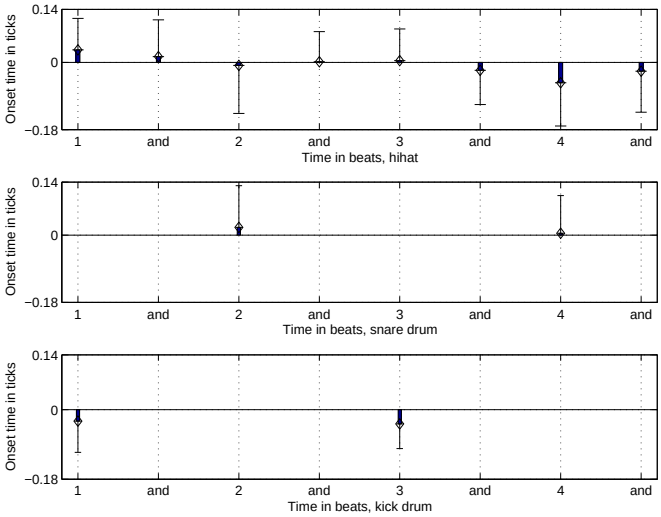


Figure 12. Onset time plot, drummer A. A rather big variance makes the groove feel less rigorous and more free and open, but this will most likely not sound very fluent when played back at high tempos. Recall that the onset time is measured in ticks between quarter notes, with range $[0 - 0.99]$.

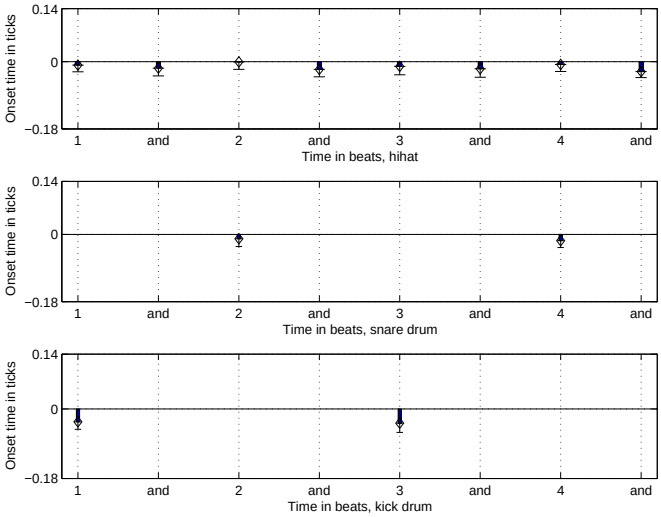


Figure 13. Onset time plot, drummer B. Constantly slightly ahead of the beat, which gives the groove a more aggressive feel.

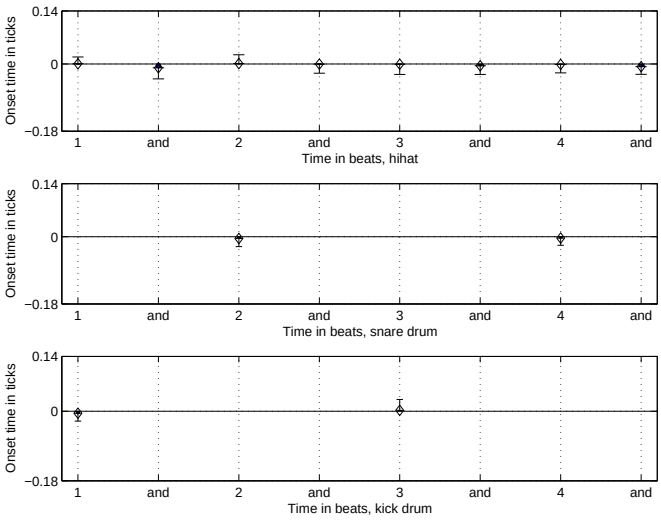


Figure 14. Onset time plot, drummer C. All the onset times are very close to the metronome, but the variations in being both before and after the beat makes this groove sound less tight than that of drummer B.

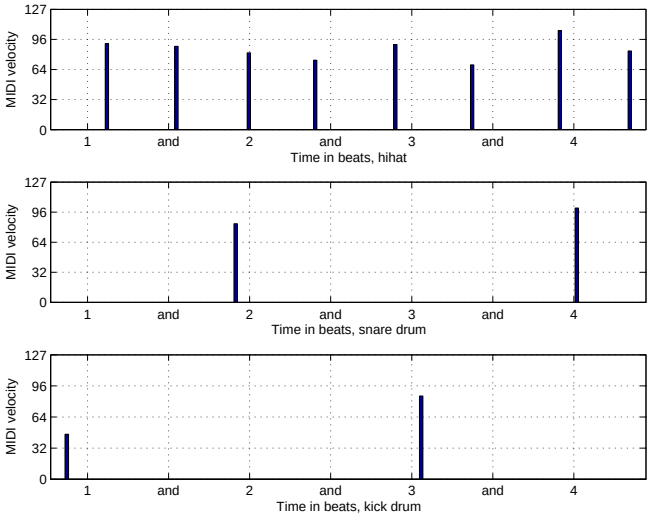


Figure 15. Imitated velocity and onset time plot, drummer A. Compare to figure 9 to see that the pattern deviates slightly from the mean and standard deviation.

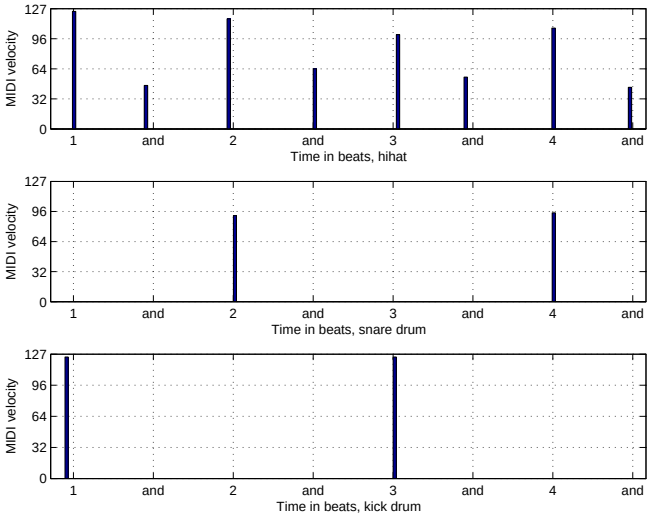


Figure 16. Imitated velocity plot, drummer B. The same "rock" feel is kept during the imitation (as can be seen in figure 10). Note how the hihat beat on the 3 is slightly behind the beat. This can be heard as a small flaw in the playing style, but will also add life to the resulting drum track.

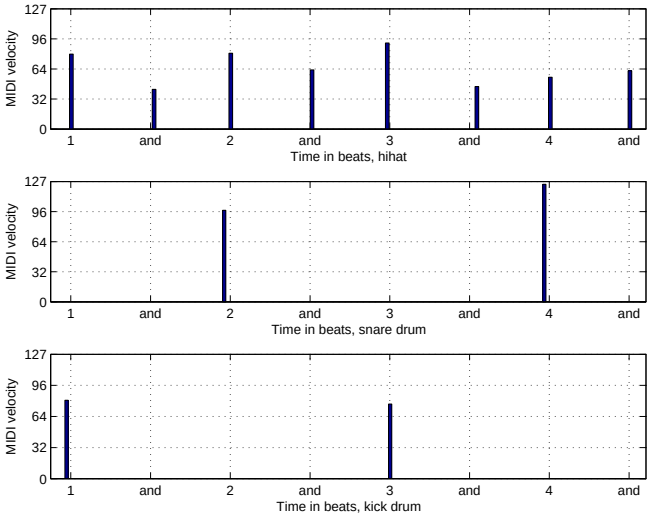


Figure 17. Imitated velocity plot, drummer C. The particular accentuated hihat beat on the 3 is present, albeit not so dominating (see figure 11 for reference). Timing is both ahead and behind the beat, as modeled.

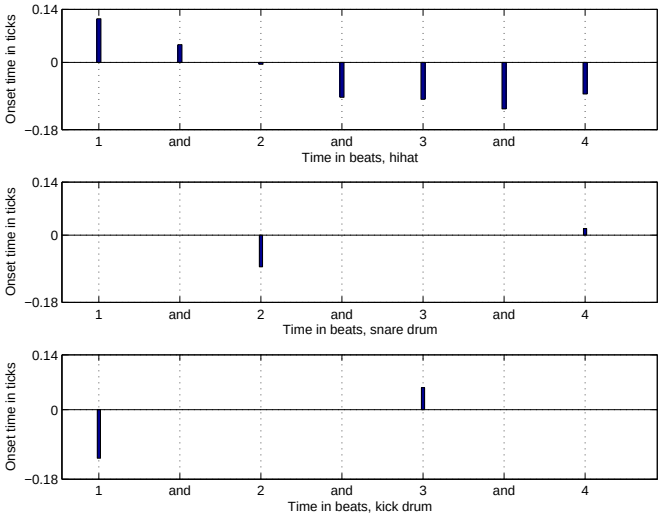


Figure 18. Imitated onset plot, drummer A. The plot complements figure 15, showing the timing with the different onset times which tend to be both ahead and behind the metronome beat.

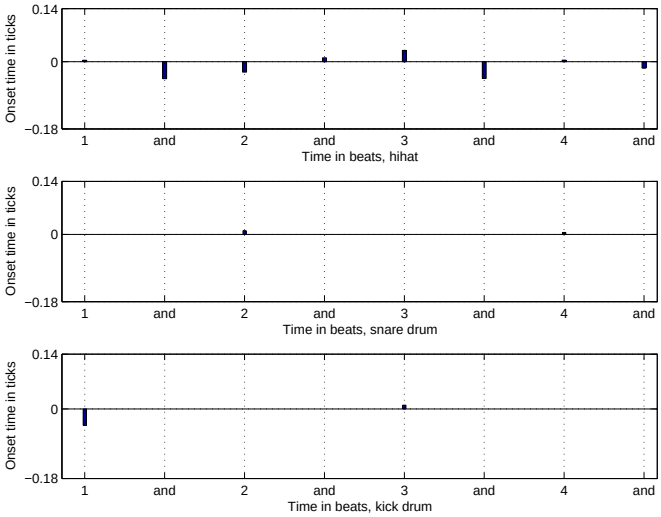


Figure 19. Imitated onset plot, drummer B. The beats are most of the time ahead of the metronome. The hihat beat on the 3 can more clearly be seen to be slightly behind the beat (as is also observable in figure 16).

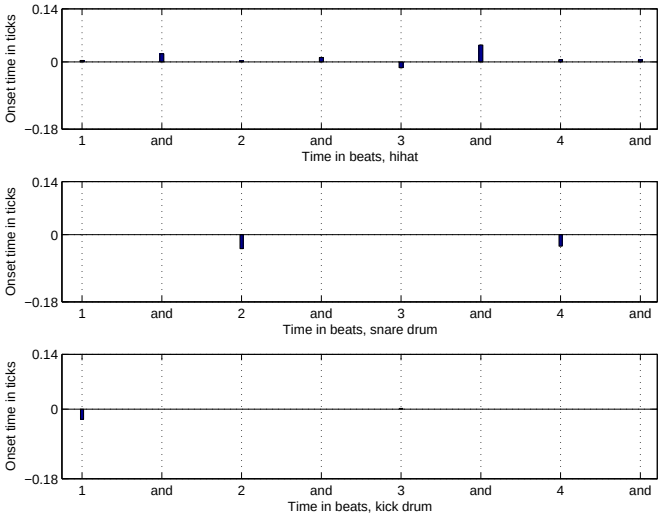


Figure 20. Imitated onset plot, drummer C. The mean was close to zero (as can be seen in figure 14); this plot clearly shows how the onset time of the beats varies both ahead and behind the beat over time.

but it can also yield drum patterns that would sound bad if played back at a tempo that is very different from when it was recorded. Another advantage of the MIDI representation is that it focuses solely on the playing style of the drummer. A drummer will often have a certain *sound* associated with him. This quality which is hard define formally is due to many factors; e.g. the brand of drums he/she is playing on, the producer, the genre of music, when it was recorded (i.e. drums recorded in the 80s sounds different from those in the 70s), to name a few. This further aids to develop the *signature* of the drummer, i.e. not just the patterns played but also the sonic qualities of the drumming. However, the results of this paper shows that human listeners are able to tell different drummers apart based only on the playing style of the drummer.

Table 2. How each drummer played the song in terms of core patterns and variations of core patterns. How each drummer originally played the song is shown to the left of each column dedicated to one drummer. How the imitated song differs from how it was originally played is shown in white text on a black background.

Drummer A		Drummer B		Drummer C	
Original	Imitated	Original	Imitated	Original	Imitated
C1	C1	C1	C1	C1	**C1V2**
C1	C1	C1	C1	C1	C1
C1	C1	C1	C1	C1	C1
C1	**C1V2**	C1V1	**C1V2**	C1	C1
C1	C1	C1	C1	C1	C1
C1	**C1V1**	C1	C1	C1V1	**C1**
C1V1	**C1V2**	C1	C1	C1	C1
C1	C1	C1	**C1V2**	C1	C1
C2	C2	C2	C2	C2	C2
C2	C2	C2V1	**C2**	C2V1	**C2**
C2	**C2V1**	C2	C2	C2	**C2V1**
C2	C2	C2	**C2V1**	C2	C2
C2V1	**C2**	C2	**C2V4**	C2V2	**C2V3**
C2	**C2V1**	C2V1	**C2**	C2V3	C2V3
C2	**C2V1**	C2	C2	C2V4	**C2**
C2	C2	C2V2	**C2**	C2V5	**C2V3**
C1	C1	C1	C1	C1	C1
C1	C1	C1	C1	C1V2	**C1**
C1V2	**C1**	C1	C1	C1V3	**C1**
C1	**C1V1**	C1V2	**C1**	C1	C1
C1	C1	C1	C1	C1	**C1V3**
C1	C1	C1	**C1V1**	C1	C1
C1	C1	C1	C1	C1V4	**C1**
C1V3	C1V3	C1V3	**C1**	C1	C1
C2	C2	C2	**C2V4**	C2	C2
C2	C2	C2	**C2V1**	C2	C2
C2V2	C2V2	C2	**C2V1**	C2	C2
C2	C2	C2	**C2V4**	C2	**C2V6**
C2V3	**C2**	C2V3	C2V3	C2V6	**C2**
C2	**C2V1**	C2	C2	C2	**C2V4**
C2V2	**C2**	C2	C2	C2V7	**C2V3**
C2V4	**C2**	C2V4	**C2V1**	C2V8	**C2**
C3	C3	C3	**C3V1**	C3	C3
C3	C3	C3	C3	C3	C3
C3	C3	C3	C3	C3	C3
C3	C3	C3	C3	C3	C3
C3	C3	C3V1	**C3**	C3	C3
C3	C3	C3	C3	C3	C3
C3V1	**C3**	C3	**C3V1**	C3	C3
C3	**C3V1**	C3	C3	C3V1	**C3**

7 Future work

One drawback of the system as it is currently implemented, is that it does not take musical context into account when modeling the different large-scale variations in a song. Very often, a drummer will make

a large-scale variation in order to highlight dynamic parts in the song or in response to other instruments' melodies. This is often referred to as *breaks* or *fills*, and can be described as being big deviations from the core pattern, e.g. playing on the toms or doing a drum roll. Currently, breaks are modeled as mere variations of a core pattern, and can be played at any point during a song. A break will typically occur only at certain places, such as the measure leading up to the chorus or to highlight a specific section of the melody. These variations should be modeled on the basis of musical context, which would aid the modeling of the other patterns as well. The current implementation of SHEILA only looks at the pattern themselves, augmenting it with musical knowledge could allow for modeling *why* a drummer would play in a specific manner in response to the melody and the dynamics of a song, i.e. *understanding* how the drummer is being creative, as attempted by Widmer [24] and Pachet [4]. In addition, if the system could handle sound input instead of MIDI files, it would give easy access to vast amounts of training data. Such a system might be implemented according to Masataka and Satoru's approach to find melody lines in pop songs, also extracting the drum pattern [11] or using one of the systems described in [9].

In addition, we are interested in modeling the physical movements of the drummer as well. Drummers play differently, not just in terms of different patterns and styles, but also in the way they move their entire body when playing. By the use of motion tracking, we aim to be able to model the physical movements of the drummer playing, which would enable SHEILA to imitate the physical playing style of a specific drummer as well. This ability could be used in a more direct multi-modal interaction setting with other musicians, and opens up another interesting field of research, namely understanding how musicians interact when playing together [21]. Work in this direction would employ the concept of using multiple forward and inverse models [14] to control the robot as it learns to imitate, as done by Demiris [6, 7]. The idea of having a library of patterns was inspired from this multiple paired models approach, however the current implementation does not use forward or inverse models.

The ability to model the style of different drummers depends on the assumption that the drums were recorded using a metronome to keep the tempo constant. However, this is often an unnatural way of playing for drummers, as the tempo becomes too rigid and is not allowed to drift in tune with the dynamics of the song. Future implementations should enable SHEILA to imitate without the assumption that the drums were recorded with a metronome, such as the approach of Cemgil et al., who uses the Bayesian framework to quantize onset times without assuming the performance was recorded using a metronome [1]. Toivainen has implemented a system that allows tracking the tempo in real-time by using adaptive oscillators [20], Desain and Honing use a connectionist approach to real-time tracking of the tempo [8]. The latter approach would be necessary if the artificial drummer would be used in a live setting, as the tempo tends to drift more than when recording in a studio.

There are a lot of interesting directions for future research, and we believe that this paper is an important first step towards building an artificial drummer.

ACKNOWLEDGEMENTS

The authors would like to thank the BioART team (Anthony Dearden, Paschalis Veskos, Matthew Johnson, Tom Carlson, Balint Takacs and Simon Butler) and the reviewers for insightful comments and constructive criticism regarding the content of this paper.

REFERENCES

[1] Ali Taylan Cemgil, Peter Desain, and Bert Kappen, 'Rhythm quantization for transcription', *Computer Music Journal*, **24**(2), 60–76, (Summer 2000).

[2] François Pachet, 'Interacting with a musical learning system: The continuator', in *ICMAI '02: Proceedings of the Second International Conference on Music and Artificial Intelligence*, pp. 119–132, London, UK, (2002). Springer-Verlag.

[3] François Pachet, 'Playing with virtual musicians: The continuator in practice', *IEEE MultiMedia*, **9**(3), 77–82, (2002).

[4] François Pachet, *Creativity Studies and Musical Interaction*, Psychology Press, 2006.

[5] François Pachet, *Enhancing Individual Creativity with Interactive Musical Reflective Systems*, Psychology Press, 2006.

[6] Yiannis Demiris and Gillian Hayes, *Imitation in animals and artifacts*, chapter Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model, 327–361, MIT Press, Cambridge, 2002.

[7] Yiannis Demiris and Bassam Khadhouri, 'Hierarchical attentive multiple models for execution and recognition of actions', *Robotics and Autonomous Systems*, **54**, 361–369, (2006).

[8] Peter Desain and Henkjan Honing, 'The quantization of musical time: A connectionist approach', *Computer Muisc Journal*, **13**(2), (1989).

[9] Simon Dixon, 'Analysis of musical content in digital audio', *Computer Graphics and Multimedia: Applications, Problems, and Solutions*, 214–235, (2004).

[10] T. Eerola and P. Toiviainen, *MIDI Toolbox: MATLAB Tools for Music Research*, University of Jyvskyl, Kopijyv, Jyvskyl, Finland. Available at http://www.jyu.fi/musica/miditoolbox/, 2004.

[11] Masataka Goto and Satoru Hayamizu, 'A real-time music scene description system: Detecting melody and bass lines in audio signals', in *Working Notes of the IJCAI-99 Workshop on Computational Auditory Scene Analysis*, pp. 31–40, (August 1999).

[12] Josep Lluís Arcos and Ramon López de Mántaras. Combining AI techniques to perform expressive music by imitation, 2000.

[13] Josep Lluís Arcos and Ramon López de Mántaras, 'An interactive case-based reasoning approach for generating expressive music', *Applied Intelligence*, **14**(1), 115–129, (2001).

[14] Michael I. Jordan and David E. Rumelhart, 'Forward models: Supervised learning with a distal teacher', *Cognitive Science*, **16**, 307–354, (1992).

[15] Ramon López de Mántaras and Josep Lluís Arcos, 'The synthesis of expressive music: A challenging CBR application', in *ICCBR '01: Proceedings of the 4th International Conference on Case-Based Reasoning*, pp. 16–26, London, UK, (2001). Springer-Verlag.

[16] Ramon López de Mántaras and Josep Lluís Arcos, 'AI and music from composition to expressive performance', *AI Mag.*, **23**(3), 43–57, (2002).

[17] Craig Saunders, David R. Hardoon, John Shawe-Taylor, and Gerhard Widmer, 'Using string kernels to identify famous performers from their playing style.', in *ECML*, eds., Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, volume 3201 of *Lecture Notes in Computer Science*, pp. 384–395. Springer, (2004).

[18] A. Tobudic and G. Widmer, 'Relational IBL in music with a new structural similarity measure', volume 2835, pp. 365–382, (2003).

[19] Asmir Tobudic and Gerhard Widmer, 'Learning to play like the great pianists.', in *IJCAI*, eds., Leslie Pack Kaelbling and Alessandro Saffiotti, pp. 871–876. Professional Book Center, (2005).

[20] Petri Toivainen, 'An interactive MIDI accompanist', *Computer Music Journal*, **22**(4), 63–75, (Winter 1998).

[21] Gil Weinberg, 'Interconnected musical networks: Toward a theoretical framework', *Computer Music Journal*, **29**(2), 23–39, (2005).

[22] Gil Weinberg and Scott Driscoll, 'Robot-human interaction with an anthropomorphic percussionist', in *CHI 2006 Proceedings*, pp. 1229–1232, (April 2006).

[23] Gil Weinberg, Scott Driscoll, and Mitchell Perry, 'Musical interactions with a perceptual robotic percussionist', in *Proceedings of IEEE International Workshop on Robot and Human Interactive Communication*, (2005).

[24] Gerhard Widmer, 'Studying a creative act with computers: Music performance studies with automated discovery methods', *Musicae Scientiae*, **IX**(1), 11–30, (2005).

A unified framework for imitation-like behaviors

Francisco S. Melo and **Manuel Lopes** and **José Santos-Victor** and **Maria Isabel Ribeiro**[1]

Abstract. In this paper, we combine the formal methods from reinforcement learning with the paradigm of imitation learning. The extension of the reinforcement learning framework to integrate the information provided by an expert (demonstrator) has the important advantage of allowing a clear decrease of the time necessary to learn certain robotic tasks. Hence, learning by imitation can be interpreted as a mechanism for fast skill transfer. Another contribution of this paper consists in showing that our formalism is able to model different types of imitation-learning that are described in the biological literature. It thus unifies in the same abstract model what used to be addressed as separate behavioral patterns. We illustrate the application of these methods in simulation and with a real robot.

1 INTRODUCTION

In the early days of behavioral sciences, several processes used by animals to acquire new skills were often dismissed as "mere imitation". As the knowledge of animal behavior, psychology and neurophysiology evolved, imitation has been promoted and is now considered a sophisticated cognitive capability that few species are capable of [1]. This change in the interpretation was accompanied by the discovery of several phenomena resulting in imitation-like behavior, *i.e.*, in a repetition of an observed pattern of behavior.

In social learning, a learner uses information provided by an *expert* to improve its own learning. For example, if the learner is able to observe the actions taken by a second subject, it can bias its exploration of the environment, improve its model of the world or even mimic parts of the other agent's behavior. This process, generally dubbed as *imitation*, makes cultural transfer of knowledge fast and reliable—acquired knowledge enables fast learning. Cultural spreading becomes thus possible by a *Lamarckian* principle, where animals learn how to act by imitating others and having the same manneriisms as their peers. Through imitation, new discoveries are learnt by each individual very efficiently, simply by observation and behavior matching.

"Real" imitation occurs when a new action is added to the agent's repertoire after having seen a demonstration. It is not enough to repeat an action after having seen it. In fact, this phenomenon can often be explained by reinforcement learning (or learning by trial-and-error). Although some social skill is usually developed when learning by trial-and-error, there is no real imitation (where *new* skills are acquired by simple observation). The concept of *imitation* is far from clear and led biologists to define several mechanisms to explain different types of *imitation-like* behaviors.

In this paper, we analyze several such imitation-like behaviors. We show how each can be modeled using a common formalism. This formalism borrows the fundamental concepts and methods from the reinforcement learning framework [2]. By considering different ways by which an expert can provide information to the learner, we feature different types of learning from observation and formalize each of the aforementioned behaviors in a reinforcement learning (RL) context.

We recall that RL addresses the problem of a decision-maker faced with a sequential decision problem and using evaluative feedback as a performance measure. The evaluative feedback provided to the decision-maker consists of a *reinforcement signal* that quantitatively evaluates the immediate performance of the decision-maker. To optimally complete the assigned task, the decision-maker must *learn* by *trial-and-error*: only sufficient exploration of its environment and actions ensure that the task is properly learnt. Therefore, in the standard RL formalism, the reinforcement signal is a fundamental element that completely describes the task to be learnt. If the agent knows how the reinforcement is assigned, it should be able to learn the task by trial-and-error (given enough time) and the information from an expert can, at most, speed up the learning process.

In real imitation as considered above, the learner should be able to acquire a new skill/learn a new task from the observations. However, and unlike the situation described in the previous paragraph, it generally should not be able to do this *without* the information provided by the expert. Considering everything stated so far, we could argue that, from a RL perspective, this corresponds to the *learning of the reinforcement function*.

Imitation has been proposed as a method to program the complex robotic systems existing today [3, 4, 5]. Programming highly-complex robots is a hard task *per se*; if a robot is capable of learning by observation and imitation, the task of programming it would be greatly simplified. To the extent of our knowledge, no systematic computational model has been proposed to formally describe imitation-like behaviors. The formalism proposed in this paper aims at fulfilling such gap. So far, the mainstream of the research in imitation aimed at individually clarifying/modeling several fundamental mechanisms individually: body correspondence [6, 7], imitation metrics [8], view-point correspondence [9] and task representation [10].

In this paper, we propose a formalism to address learning from observations. In this formalism, several types of information provided by an expert are integrated in a RL framework in different ways. We consider different assumptions on the information provided and on the way this information is integrated in the learning process, and show that this results in different imitation-like behaviors. It is our belief that the formal approach in this paper contributes to disambiguate several important concepts and clarify several issues arising in the literature on learning by imitation.

The paper is organized as follows. Section 2 reviews the main concepts in imitation learning. We describe several models of imitation in biological and artificial systems, as well as some computational problems arising in the context of imitation. Section 3 describes the framework of RL and introduces the fundamental notation. We pro-

[1] Institute for Systems and Robotics,Instituto Superior Técnico, Lisboa, Portugal.*E-mail*: {fmelo, macl, jasv, mir}@isr.ist.utl.pt

ceed in Section 4 by analyzing several methods to use expert information to speed learning. We show these methods to fall within specific classes of imitation-like behavior. To do this we describe how imitation and reinforcement can be combined and describe two simple methods to achieve this. We illustrate some of the methods in the paper in Section 5 and conclude the paper in Section 6.

2 IMITATION LEARNING

Several different mechanisms can result in a imitation-like behavior. One agent may perform an action after having seen it, but the mechanisms leading to it may be very different. Even when asking someone to imitate a hand movement, the results may vary substantially depending on the individual in question [11, 12]. From the study of imitation in animals, several mechanisms were proposed to describe an "imitative behavior", [1, 13, 4, 3]:

1. **Stimulus Enhancement** describes the general tendency to respond more vigorously toward those parts of the environment within which a conspecific is seen to interact. Seeing what are the important parts of the environment and which objects might be useful can speed up learning;
2. **Contextual Learning** describes the situation when an action is not learned, but the perception of a new object property can produce the desire to act upon it. If, for example, an animal sees someone throwing a coconut, it will learn the possibility of throwing it. In the context of our work, contextual imitation would amount to learning to employ an action, already known, in different circumstances.
3. **Response facilitation** is described in [1] as "a kind of social effect that selectively enhances responses: watching a conspecific performing an act, often resulting in a reward, increases the probability of an animal doing the same." Large flocks of birds fly in perfect synchronization. They are not imitating each other, but simply doing the same to protect themselves against predators.
4. **Emulation** can also lead to a behavioral match. Observing an action and the corresponding result might bring a desire to obtain the same goal. Learning that a coconut can be smashed to reach the inside will give the desire to eat the inside and thus producing the same behavior.

Although the mechanisms just described produce imitative behavior, they do not exactly correspond to imitation learning, in the sense that no "new actions" are learned from scratch or added to the existent repertoire. On the other hand, there is a second set of processes leading to imitative behavior where learning of new actions does actually occur. This is called *production learning* [13] and, as it is the most-powerful way of imitation, the "true-imitation" [3].

Byrne distinguishes two cases of production learning, namely **action-level** and **program-level** learning [13]:

- **Action-level learning** is defined as: "The indiscriminate copying of the actions of the teacher without mapping them onto more abstract motor representation." [3]. This is a perfect copy of the motions, if the kinematics of the systems are the same, even the joint level trajectories are the same.
- **Program-level learning** defined for the cases where not only the superficial motion is copied but when a broader description of the sequences, goals and the hierarchical structure of the behavior is inferred by the learner [14].

From the examples above we can see that many situations dubbed as imitation do not involve any actual learning, but only simultaneous/similar action. Response facilitation is just the equal answer that similar agents give when they are at the same state. Emulation and contextual learning can be explained as an improvement of the world model. The result of some action, or its relevant use in a given situation is added to the possible actions. In stimulus enhancement some task learning occurs, but the action is learned by trial-and-error, the demonstration only providing partial knowledge. In imitation, we expect the agent to learn how to complete the task or even the task itself.

2.1 Some implementation issues

Imitation cannot be reduced to supervised learning, where the agent is given the input and correct output. In imitation, the agent is given a set of observations of the environment and corresponding adequate actions. It must then *translate* this information *in terms of its own body*. This is the first difficulty in imitation: the observation is made from a different point-of-view. The different actions performed then must be *recognized* and *mapped to the agent's different capabilities*. Finally, the agent must *infer the important parts of the demonstration*. In imitation, all these problems must be carefully addressed, this being the reason why imitation is considered a complex cognitive task. We now discuss each of these three steps in detail.

Due to the problem of "seeing the world from another's viewpoint", the observed actions must be translated into the referential frame of the learner due to the different perceptual viewpoints, *i.e.*, the learner must perform a "mental rotation" to place the demonstrator's body (*allo-image*) in correspondence with the learner's own body (*ego-image*) [15, 9, 16].

Furthermore, when considering the problem of learning by imitation there is some *correspondence* assumed between the body of the demonstrator and that of the imitator. The *correspondence problem* is precisely defined as the mapping between the actions, states and effects of the demonstrator and those of the imitator. It is particularly relevant if the actions are performed by a specific body and should be replicated by a different body. Even when considering similar bodies, contextual knowledge or training may imply that the demonstrator and the imitator cannot use one same object in the same ways. And if this is not the case, there are always small differences in kinematics, size, dynamics or context that require the correspondence problem to be solved. This problem can be addressed using different methodologies. Examples include algebraic approaches [17], trajectory balance correction [18] and matching the effects of the actions [7].

Finally, it is necessary to *evaluate* the performance of the imitator. In other words, an agent needs a metric that, in a sense, allows it to determine if the imitation was successful or not. And, as expected, different metrics can will lead to different results. These *imitation metrics* evaluate how well the imitator was able to grasp underlying goal of the demonstrated task. These metrics can be selected using an algebraic formulation [8], by optimizing the learnt trajectories [19] or by considering the visual process involved [9].

Figure 1 combines the previous elements in an illustrative architecture that summarizes the relation between these elements of imitation learning [5]. In this paper we do not address the fundamental problems of view-point transformations or recognition. Instead, we assume that the learner receives the processed output of the blocks computing the VPT and performing the recognition, and focus in the problem of learning.

As will soon become apparent, we provide a unified framework to address imitation learning and reinforcement learning. We show that, in this setting, there is an imitation metric that arises naturally from the formulation of the problem of imitation. Furthermore, we describe several situations where such metric does not arise naturally from the problem formulation. We identify in each such situation

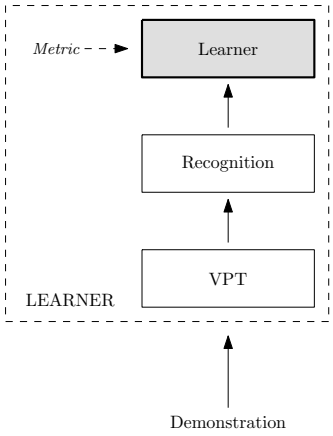


Figure 1. Architecture for the imitator.

a particular instance of *imitation-like behavior*, where the learning agent "appears" to imitate the demonstrator but where no actual imitation takes place.

3 REINFORCEMENT LEARNING

The general purpose of RL is to find a "good" mapping that assigns "perceptions" to "actions". Simply put, this mapping determines how a decision-maker reacts in each situation it encounters, and is commonly known as a *policy*. The use of evaluative feedback, by means of a *reinforcement signal*, allows the decision-maker to gradually grasp the *underlying purpose* of the task it must complete while optimizing the way of completing it.

In this section we describe *Markov decision processes*, the standard framework used to address RL problems. We also review some solution methods that we later employ in the context of imitation.

3.1 Markov decision processes

Let $\{X_t\}$ denote a controlled Markov chain, where The parameter t is the discrete time, and X_t takes values in a finite set $\mathcal{X}$, known as the *state-space*.

The distribution of each r.v. X_{t+1} is conditionally dependent on the past history $\mathcal{F}_t$ of the process according to the probabilities

$$\mathbb{P}\left[X_{t+1} = y \mid \mathcal{F}_t\right] = \mathbb{P}\left[X_{t+1} = y \mid X_t = x, A_t = a\right] = \\ = \mathsf{P}_a(x, y).$$

We note that the transition kernel P depends at each time instant t on a parameter A_t, which takes values in a finite set $\mathcal{A}$. This parameter provides a decision-maker with a mechanism to "control" the trajectories of the chain by influencing the corresponding transition probabilities. We generally refer to the sequence $\{A_t\}$ as the *control process*; we refer to A_t as the *action at time instant* t and to $\mathcal{A}$ as the *action-set*.

Every time a transition from a state $x \in \mathcal{X}$ to a state $y \in \mathcal{X}$ occurs under a particular action $a \in \mathcal{A}$, the decision-maker is granted a numerical *reinforcement* $r(x, a, y)$. This reinforcement provides the evaluative feedback that the decision-maker must use to learn the desired task. The decision-maker must determine the control process $\{A_t\}$ maximizing the *expected total discounted reward*, as given by the functional

$$J(\{A_t\}, x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x\right],$$

where we denoted by R_t the reinforcement received at time t, given by $r(X_t, A_t, X_{t+1})$. Throughout the paper, we admit that the rewards are bounded, *i.e.*, , $|r(x, a, y)| \leq \mathcal{R}$ for some constant $\mathcal{R}$. Also, and to simplify the discussion, we admit r to be constant on the second and third parameters. The parameter $0 < \gamma < 1$ is a discount factor.

A *Markov decision process* (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where $\mathcal{X}$ is the state-space, $\mathcal{A}$ is the action-space, P represents the transition probabilities for the controlled chain and r is the reinforcement function.

3.2 Dynamic programming and stochastic approximation

We define a *policy* as being a state-dependent decision-rule, and denote it as a mapping $\delta_t : \mathcal{X} \times \mathcal{A} \longrightarrow [0, 1]$ assigning a probability $\delta_t(x, a)$ to each state action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$. The value $\delta_t(x, a)$ represents the probability of $A_t = a$ when $X_t = x$. A policy δ independent of t is dubbed as *stationary*, and as *deterministic* if for each $x \in \mathcal{X}$ there is an $a \in \mathcal{A}$ such that $\delta_t(x, a) = 1$. In the latter case, we abusively denote by $\delta_t(x)$ the action determined by δ_t when $X_t = x$.

The *value function* associated with a policy δ_t is defined as a mapping $V^{\delta_t} : \mathcal{X} \longrightarrow \mathbb{R}$ defined for each state $x \in \mathcal{X}$ as

$$V^{\delta_t}(x) = J(\{A_t\}, x),$$

where the control process $\{A_t\}$ is generated from $\{X_t\}$ according to δ_t. Given an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, there is at least one deterministic, stationary policy δ^* such that

$$V^{\delta^*}(x) \geq V^{\delta_t}(x),$$

for any policy δ_t and any state $x \in \mathcal{X}$. This policy can, in turn, be obtained from V^{δ^*} as

$$\delta^*(x) = \arg\max_{a \in \mathcal{A}} \Big[r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) V^{\delta^*}(y)\Big].$$

Any such policy is *optimal* and the corresponding value function V^{δ^*} is simply denoted by V^*. Clearly, V^* verifies the recursive relation

$$V^*(x) = \max_{a \in \mathcal{A}} \Big[r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) V^*(y)\Big],$$

known as the *Bellman optimality equation*. Notice that $V^*(x)$ is the expected total discounted reward along a trajectory of the Markov chain starting at state x obtained by following the optimal policy δ^*.

From V^* we define a function $Q^* : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$ as

$$Q^*(x, a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) V^*(y).$$

The value $Q^*(x, a)$ is the expected total discounted reward along a trajectory of the chain verifying $X_0 = x$ and $A_0 = a$, obtained by following the optimal policy for $t \geq 1$.

Summarizing, we have the following relations

$$V^*(x) = \max_{a \in \mathcal{A}} Q^*(x, a); \tag{1a}$$

$$Q^*(x, a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) \max_{b \in \mathcal{A}} Q^*(y, b); \tag{1b}$$

$$\delta^*(x) = \arg\max_{a \in \mathcal{A}} Q^*(x, a). \tag{1c}$$

Now given any functions $v : \mathcal{X} \longrightarrow \mathbb{R}$ and $q : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$, we consider the operators

$$(\mathbf{T}v)(x) = \max_{a \in \mathcal{A}} \Big[r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x,y) v(y) \Big]$$

and

$$(\mathbf{H}q)(x,a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x,y) \max_{b \in \mathcal{A}} q(y,b).$$

It is straightforward to see that V^* and Q^* are fixed points of the operators $\mathbf{T}$ and $\mathbf{H}$. Each of these operators is a contraction in a corresponding norm and thus a simple fixed-point iteration can be used to determine V^* and Q^*.

The use of either $\mathbf{T}$ or $\mathbf{H}$ to determine V^* or Q^* by fixed-point iteration is a process known as *value iteration*. It is a dynamic programming approach that is often used to determine the function V^* and Q^* from which the optimal policy δ^* can be computed.

When this is not the case, *i.e.*, when P and r are unknown, many methods have been proposed that asymptotically converge to the desired functions [20, 2]. In this paper, we use use one of the most studied methods in the RL literature: the Q-learning algorithm [21]. This method uses sample trajectories of the Markov process, $\{x_t\}$, control process, $\{a_t\}$ and corresponding rewards $\{r_t\}$ to estimate the function Q^*. These estimates are updated according to the Q-learning update

$$\begin{aligned} Q_{t+1}(x_t,a_t) = (1 - \alpha_t(x_t,a_t)) Q_t(x_t,a_t) + \\ + \alpha_t(x_t,a_t) \big[r_t + \gamma \max_{b \in \mathcal{A}} Q_t(x_{t+1},b) \big]. \end{aligned} \tag{2}$$

This algorithm will converge to Q^* w.p.1 as long as $\sum_t \alpha_t(x,a) = \infty$ and $\sum_t \alpha_t^2(x,a) < \infty$ for every $(x,a) \in \mathcal{X} \times \mathcal{A}$. This requires in particular that every state-action pair be infinitely often (there is sufficient exploration of the environment and the agent's actions).

4 LEARNING PARADIGMS USING EXPERT INFORMATION

In the previous sections we described two learning paradigms: learning by imitation and learning by reinforcement. In this section we move towards a combined learning framework, the *learning by observation and reinforcement* (LOR) framework. To this purpose, we consider a learning agent that must learn how to perform a sequential task using some prior knowledge and information provided by an expert.

The formalism considered herein borrows the fundamental ideas from the reinforcement learning framework described in the previous section, thus providing a unified framework to address both classes of learning processes. The fundamental assumptions usually considered in the reinforcement learning framework are:

- The task to be learnt can be described as a mapping from the set of states of the environment to the set of possible actions (a *policy*);
- The environment is *stationary*.

The first assumption simply states that in the same state of the environment the agent should always perform the same action. We remark that this assumption bears yet another important implication. If, as stated, the task to be accomplished can be fully described using a policy, then there is a reward function such that the *policy to be learnt is the optimal policy with respect to this reward function*, in the sense described in Section 3.

The second assumption above simply means that the policy used to fulfill the task should not change with time (the environment always responds to the agent's actions in the same way).

In what follows, we will consider two fundamental situations:

(i) The imitator knows the task to be learnt, but does not know how to perform this task;
(ii) The imitator does not know the task to be learnt.

From everything stated so far, it should be clear that, in terms of our formalism, the situation in (i) simply means that there is a previously defined reward function, known by the agent (since the reward function defines the task). Notice that if the agent is aware of this function, it can learn to perform the task by trial-and-error, given sufficient time. Clearly, the situation in (ii) means that *there is no reward function defined a priori*. This, of course, implies that the agent will not be able to learn any task without any further information.[2]

We analyze how different types of information provided by an expert can be integrated in learning the desired task. As will soon become apparent, models for the imitation-like behaviors described in Section 2 arise naturally in the LOR framework. We also show that, in the more complex scenario of an unknown task, it is possible to provide a natural interpretation for the used algorithm in terms of imitation metrics. The first case do not correspond to *true-imitation* behavior as the system already knows the task, only in the second case we can talk about *true-imitation*, in the end of this section we discussion this.

We consider each of the two situations (i) and (ii) in Subsections 4.1 and 4.2, respectively.

4.1 Known task

We consider that the interaction of the learning agent and the environment can be described as a controlled Markov chain, as in Section 3. This means that, at each time t, the state of the environment will move from a state $X_t = x$ to a state $X_{t+1} = y$ depending on the action A_t of the agent and according to the transition probabilities $P_a(x,y)$. We suppose that an expert provides the learning agent with some information on how the task can be completed. We refer to such information generally as a *demonstration* and analyze how can this information be used in the learning process. We consider four distinct cases:

(i) The demonstration consists of a sequence of states,

$$\mathcal{H} = \{x_1, \ldots, x_N\},$$

obtained by following the optimal policy;
(ii) The demonstration consists of a sequence of state-actions pairs,

$$\mathcal{H} = \{(x_1,a_1), \ldots, (x_N,a_N)\},$$

"hinting" on which should be the optimal action a_i at each state x_i visited;
(iii) The demonstration consists of a sequence of transition triplets,

$$\mathcal{H} = \{(x_1,a_1,y_1), \ldots, (x_N,a_N,y_N)\},$$

providing the imitator with information on the behavior of the environment;

[2] We could argue that the situation in (ii) means that the agent *does not know* the reward function, but that the latter is defined. We do not adopt such position for the simple reason that, if a reward function *is* defined, the agent can still learn by trial-and-error and, therefore, there is no significative difference from (i).

(iv) The demonstration consists of a sequence of transition-reward tuples,

$$\mathcal{H} = \{(x_1, a_1, r_1, y_1), \ldots, (x_N, a_N, r_N, y_N)\},$$

providing the imitator with information on the behavior of the environment and on how the task should be completed.

First of all, we remark that, since we assume knowledge of r, (iii) and (iv) are redundant. Nevertheless, we will consider how to address the two situations distinctly, noting that in (iv) allows to address situations in which r is unknown.

We must further detail the idea behind each of the previous classes of demonstrations. The first situation, (i), addresses situations in which the learning agent *can not observe/recognize the actions of the demonstrator* but only their effect in the environment. This information will show the agent how the state of the environment should evolve when the optimal policy is implemented. A sequence as described in (ii) illustrates *how the task can be completed*. Each pair (x_i, a_i) is related through some deterministic policy δ that is "close" to optimal. Sequences as those described in (iii) and (iv) illustrate *the dynamics of the environment* in terms of transitions and transitions-rewards, respectively. Unlike the sequences described in (ii), it is not assumed that x_i and a_i in each tuple (x_i, a_i, y_i) or (x_i, a_i, r_i, y_i) are related by some policy.[3]

Another important aspect is that, at this stage, we are not concerned with the particular way by which the sequences $\mathcal{H}$ in (ii) through (iv) are obtained. Consider for example the situation in (ii). It may occur that the demonstrator illustrates how the task is completed by demonstrating the action to be chosen in an arbitrary set of states $\{x_1, \ldots, x_N\}$. Or, it may happen that the sequence of states $\{x_1, \ldots, x_N\}$ is actually a sample path of the process obtained with the control sequence $\{a_1, \ldots, a_{N-1}\}$.

We also remark that, in all 3 cases listed above, we assume that the imitator is able to perceive the information in the sequences $\mathcal{H}$ unambiguously. We could admit *partial observability*, meaning that the imitator was able to observe the states, actions and/or rewards in the sequences $\mathcal{H}$ only up to some degree of accuracy. This would imply that the imitator would have to *estimate* what the actual state, action and/or reward would have been. This, of course, would be the actual case in practical situations. Nevertheless, consideration of partial observability adds no useful insight to our formalization of the imitation problem and significantly complicates the presentation.

The four methods presented below all provide an initial estimate Q_0 for Q^* that integrates the information provided by the demonstration. We will see that this informed initialization brings a significant improvement in the learning performance of the agent.

Method 4.1.1: Sequence of states

Consider a sequence of states

$$\mathcal{H} = \{x_0, \ldots, x_N\},$$

obtained according to the optimal policy. As stated, this first scenario comprises situations where the learning agent is not able to observe/recognize the actions performed by the expert. Nevertheless, the sequence of states $\mathcal{H}$ provides the learning agent with an idea on how the environment evolves "under" the optimal policy.

[3] We make this distinction as each of the sequences described in (i) through (iv) provides the imitator with different information, to be used in different ways. This is not limiting in any way, as discussed below.

Therefore, if the transition model is known, the agent can compute

$$Q_0(x, a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) V^*(y),$$

where V^* is computed as $V^* = (\mathbf{I} - \gamma \mathsf{P}^*)^{-1} r$. The matrix $\mathbf{I}$ denotes the identity and the transition matrix P^* represents the transition model for the optimal policy, estimated from $\mathcal{H}$ as

$$\mathsf{P}^*(x, y) = \frac{N(x, y)}{\sum_{z \in \mathcal{X}} N(x, z)},$$

where $N(x, y)$ denotes the number of times that a transition from x to y occurred in $\mathcal{H}$. This method is similar to that proposed in [22].

Method 4.1.2: Sequence of state-action pairs

Consider a sequence of state-action pairs

$$\mathcal{H} = \{(x_1, a_1), \ldots, (x_N, a_N)\}.$$

Each demonstrated pair (x_i, a_i) provides significant information on the *optimal policy* at x_i. And even if the policy partially defined by $\delta(x_i) = a_i$ is not optimal, it is expectable that it is "close" to optimal. It is therefore reasonable that the imitator *uses δ as an initial policy to perform the task*. And, as it acquires further experience on the task, it should be able to improve from this initial policy, if there is room for such improvement. To incorporate this information in the initial estimate for Q^*, we set $Q_0(x_i, a_i) = 1$ for $i = 1, \ldots, N$ and 0 otherwise.

Method 4.1.3: Sequence of transition triplets

We now consider a sequence of transition triplets

$$\mathcal{H} = \{(x_1, a_1, y_1), \ldots, (x_N, a_N, y_N)\}.$$

As mentioned above, this sequence provides the imitator with *information on the behavior of the environment*. Clearly this is only useful if the transition probabilities are not known *a priori*. If this is the case, the information provided by the demonstrator can be used to improve the model of the environment by setting

$$\hat{\mathsf{P}}_a(x, y) = \frac{N(x, a, y)}{\sum_{z \in \mathcal{X}} N(x, a, z)},$$

where $N(x, a, y)$ denotes the number of times that the triplet (x, a, y) was observed in $\mathcal{H}$. This estimated transition model $\hat{\mathsf{P}}$ with the function r can be used to perform value iteration and obtain an initial estimate Q_0 for the learning algorithm.

Method 4.1.4: Sequence of transition-reward tuples

Finally, we consider a sequence of transition-reward tuples

$$\mathcal{H} = \{(x_1, a_1, r_1, y_1), \ldots, (x_N, a_N, r_N, y_N)\}.$$

This sequence provides the imitator with information on the behavior of the environment and on *the task*. This means that the tuples in $\mathcal{H}$ can be used to perform N iterations of Q-learning using (2). The resulting Q-function provides the initial estimate Q_0 for the learning algorithm.

4.2 Unknown task

In this subsection, we use the exact same formulation considered in Subsection 4.1 above, but suppose that *no reward mechanism is defined*. This means that the imitator is no longer able to learn the task by trial-and-error if no demonstration is available.

However, if a demonstrator provides the imitator with some information on how the task can be completed, the imitator can *build* its own reward function and use it to learn how to perform the task. We also refer to such information generally as a *demonstration*.

Unlike in the previous situation, we only consider two scenarios: We consider four distinct cases.

(i) The demonstration consists of a sequence of states,

$$\mathcal{H} = \{x_1, \ldots, x_N\}\,.$$

(ii) The demonstration consists of a sequence of transition triplets,

$$\mathcal{H} = \{(x_1, a_1, y_1), \ldots, (x_N, a_N, y_N)\}\,,$$

providing the imitator with information on the behavior of the environment.

Notice that, since there is no reward function defined, it is not possible to consider the situation where transition-reward tuples are observed. Also, and unlike Subsection 4.1, we now assume that the transition triplets in $\mathcal{H}$ considered in (ii) are obtained *using the policy to be learnt*. Therefore, (ii) includes both (ii) and (iii) from the previous subsection.

Method 4.2.1: Sequence of states

Consider a sequence of states

$$\mathcal{H} = \{x_0, \ldots, x_N\}\,,$$

obtained according to the optimal policy. As in Subsection 4.1, this scenario comprises situations where the learning agent is not able to observe/recognize the actions performed by the expert.

We interpret the sequence of states in $\mathcal{H}$ as providing the learner with information *on the goal* of the task. In particular, we consider that $\mathcal{H}$ represents a *possible trajectory to a goal state*. Therefore, the learner will memorize the last state visited, x_N, as the goal state and build a simple reinforcement function defining the task "reach the goal state as fast as possible". An example of one such reward function is

$$r(x) = \begin{cases} +10 & \text{if } x = x_N; \\ -1 & \text{otherwise.} \end{cases}$$

The agent can now apply any preferred method to determine the optimal policy. For example, it can use value iteration if P is known, or Q-learning otherwise. The learner will thus learn a policy that will partially replicate the demonstration observed.

Method 4.2.2: Sequence of transition triplets

We now consider a sequence of transition triplets

$$\mathcal{H} = \{(x_1, a_1, y_1), \ldots, (x_N, a_N, y_N)\}$$

obtained using the "optimal policy". As in Subsection 4.1, this sequence can be used to improve the model of the environment. This model of the environment can, in turn, be used to determine the reward function that best translates the policy partially defined by $\delta(x_i) = a_i$, $i = 1, \ldots, N$. The approach considered here differs from that used in Method 4.2.1 in that the reward function is no longer built by considering only one final state. Instead, the learning agent will use the *whole demonstration* and apply inverse reinforcement learning to build the reward function [23]. We will show that this procedure is fundamentally different from the previous ones, and corresponds to "real imitation" in the sense of Section 2.

4.3 Classification of the learning paradigms

So far in this section we formalized several different methods by which an agent can use the information provided by an expert in learning how to accomplish a task. However, as discussed in Section 2, there are several learning paradigms that do exhibit imitative behavior but which cannot be truly classified as "imitation". And, as we show in the continuation, most of the methods described above actually fall in one of the following categories:

- Stimulus enhancement;
- Contextual learning;
- Response facilitation;
- Emulation.

We start with the Method 4.1.1. In this method, the learning agent seeks to replicate the *effect* of the actions of the demonstrator. This will actually lead to an initial replication of the demonstrator's policy, but the process by which this behavioral match is attained is *emulation*.

In Method 4.1.2, the imitator uses the demonstration to *bias its learning strategy*. Therefore, this method is actually a *stimulus enhancement* mechanism: the imitator observes some actions that can be useful for the task and uses this information to speed learning.

In Method 4.1.3, the imitator uses the demonstration to *improve its model of the world*. This means that the imitator gains further knowledge on what the consequences of some of its actions may be. We can classify this as a subtle form of *contextual learning*.

A similar thing occurs in Method 4.1.4. In this method, however, the imitator further observes *the rewards* obtained by the imitator. It realizes not only the consequences of some actions but also on *how these actions contribute to complete the task*. This use of the reward information allows us to realize that Method 3 combines contextual learning with *response facilitation*.

Notice that, in all these methods, the agent already knows the task to be learnt. This means that, with enough time, the agent could learn the task without any help from a demonstrator. Furthermore, independently of the policy used in the demonstration, the agent will eventually learn the correct policy, completely disregarding the demonstration if necessary. This means that the demonstration only provides a means for the agent to speed up its own learning process. Therefore, it is not surprising that all these situations do not correspond to "true-imitation" behaviors.

Moving to the the methods in Subsection 4.2, we start by noticing that, in Method 4.2.1 the agent seeks to replicate the final *effect* of the actions of the demonstrator. In fact, in this method, the agent focuses all its learning in *replicating the effect* observed in the demonstration (in terms of final state), displaying a flagrant example of *emulation*.

On the other hand, Method 4.2.2 seeks to *extrapolate the task behind the actions of the demonstrator*. From this information, the agent builds a reward function that will eventually lead to a replication of the demonstrator's policy. However, the actual method for computing this reward function (and, thus, realizing the task to be learnt) provides important insights into the problem of imitation, that we discuss next.

4.4 Inverse reinforcement learning and imitation metrics

As argued in Section 2, "true" imitation will occur if a broad description of the action sequences, goals and hierarchical structure of the desired behavior is inferred by the learner. As we have seen, in the RL formalism, the goals and structure of the desired behavior are "encoded" in the reward function. Therefore, learning the reward function and using it to determine the optimal policy would fit the above description of true imitation.

Notice that we consider Method 4.2.1 to be emulation because two completely different sequences ending in a common final state will lead the learning agent to infer the exact same reward function. This means that, as stated in the previous subsection, this method seeks to replicate the effect of the actions of the demonstrator rather than to extrapolate the task behind the actions of the demonstrator.

On the other hand, Method 4.2.2 does seek to extrapolate this information from the demonstration. To better realize how this method operates, we provide a brief description of its working [23].

Given the model of the environment (namely the transition probabilities in P), the inverse reinforcement learning method used (dubbed *Bayesian inverse reinforcement learning*—BIRL) searches the space of possible reward functions. To this purpose, the method considers a fine discretization of the referred space of reward functions. Then, given any initial reward function, the method evaluates the optimal Q-function, Q^*, for this reward function and evaluates the *likelihood of the demonstrated policy being optimal* given Q^*. This likelihood also takes into consideration a numerical parameter describing the *confidence on the optimality of the demonstrated policy*. The method will thus output the most likely reward given the demonstrated policy (obtained from $\mathcal{H}$) and the confidence parameter.

We emphasize several important aspects of this approach. First of all, this method considers the demonstration *as a whole*, instead of focusing on particular aspects. Therefore, the reward thus determined will more accurately the task "behind" the demonstration. On the other hand, the likelihood function used to compare different reward functions as well as the confidence parameter naturally provide an imitation metric for the problem. The inclusion of the confidence parameter is an important aspect that allows the agent to realize how strict it should follow the provided demonstration. A low confidence parameter will result in a learnt policy significatively more different from the demonstrated policy than a high confidence parameter.

Also notice that considering imitation metrics makes no sense in the other methods. In the methods in Subsection 4.1 the demonstration is only used to speed the learning. The agent is not trying to replicate the demonstration but to optimize its policy with respect to the pre-defined rewards. In Method 4.2.1, on the other hand, the agent is simply trying to reach the final state observed in the demonstration. Once again, is not trying to replicate the demonstration but to optimize the policy leading it to this goal state.

The reward function thus constructed will provide adequate evaluative feedback on the task and the imitator can use this evaluative feedback to optimize its own policy. We emphasize that, without the demonstration, the imitator *has no knowledge on the task*. The reward function built from the demonstration is, therefore, *new knowledge* that describes the task at hand and allows the imitator to learn how to perform it in an optimal fashion.

4.5 Discussion

With the methods above we conclude the presentation of the LOR framework. Within this framework, we model an agent's environment as a controlled Markov chain $\{X_t\}$. The demonstration provided by an expert is, in turn, described as a sequence $\mathcal{H}$ which can take various forms, depending on the information provided. The formalism considered herein borrows fundamental ideas from reinforcement learning and provides a unified framework to address both classes of learning processes.

We notice that the MDP model considered in this paper is the simplest model used in reinforcement learning. We are interested in establishing a unified framework to address both learning by imitation and reinforcement and thus focus on this simpler model for the sake of clarity. In Section 6 we briefly comment on how the fundamental framework considered herein can be extended to accommodate richer RL models (such as POMDPs).

As argued in Section 2, imitation cannot be reduced to supervised learning and, therefore, the framework presented here should not be seen as simple a combination of supervised learning and reinforcement learning.[4] Instead, it should be seen as a formalism to describe learning processes in which imitation and reinforcement learning can be properly modeled.

It is possible to find other works in the literature that combine learning by imitation and reinforcement. In [22], imitation arises *implicitly* in non-interactive multiagent scenarios. In it, a learning agent uses the trajectories observed from other agents to speed the learning of its individual task (which is generally independent of that of the others). In yet another example, [25], a learning method is proposed that learns a reinforcement function and dynamic model from the demonstration of an expert (human executer). This is then combined with a model-free, task-level direct learner to compensate for modeling errors.

Our work is fundamentally different from those considered above in that our aim is to understand how can the problem of imitation be modeled and how can imitative-like behaviors be distinguished with a formal perspective. Nevertheless, several methods described in our paper can be seen as simplified versions of the methods described in those papers.

Also, as argued in Section 2, we considered that in order for the learning mechanism to be properly classified as *imitation*, it should be able to *realize* the task from the demonstration. However, it should be flexible enough to feature two possible behaviors: to *replicate* the exact behavior of the demonstrator or, instead, to *perceive* the purpose of the task and, eventually, optimize beyond whatever it observed. As discussed in the previous subsection, the use of Method 4.2.2 verifies all these requisites. On the other hand, each of the remaining methods exhibits one of the above features, not all. This is the reason why we classified them as imitation-like.

Finally, we remark that the classical inverse reinforcement learning algorithms [26, 27] also determine a reward function given a policy. The difference from these methods to the one used here is that BIRL allows the policy to be only *partially specified* and *suboptimal*. This is an important advantage in the problems considered herein.

5 EXPERIMENTS

We conducted several simple experiments to evaluate the performance of proposed methods against that of simple trial-and-error. We evaluated each of the methods described in Section 4.

[4] Such approach is adopted, for example, in [24], where a supervisor is combined with an actor-critic learning architecture.

The task considered is a simple recycling game, where a robot must separate the objects in front of him according to its shape (Fig. 2). In front of the robot are two slots (Left and Right) where 3 types of objects can be placed: Large Ball, Small Ball and Box. The boxes should be dropped in the corresponding container and the small balls should be kicked out of the table. The large balls should be touched upon. Every time a large ball is touched, all objects are removed from the table.

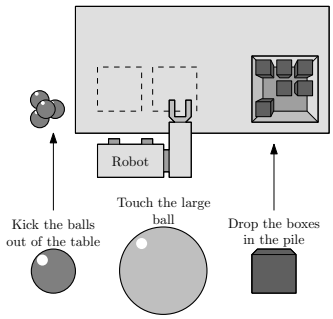


Figure 2. Simple recycling game.

The robot has, therefore, 6 possible actions: Touch Left (TL), Touch Right (TR), Kick Left (KL), Kick Right (KR), Grasp Left (GL) and Grasp Right (GR). We notice that, if the robot kicks a ball on the right while an object is lying on the left, the ball will remain in the same spot. The robot receives a reward of $+10$ every time the table is empty and -1 every other time.

The correct policy for this game is to touch the large ball, if there is any, or get rid of the object on the left and then of the object on the right (there are some situations where the order is not important). Every time the table is emptied, the game is restarted.

We tested the performance of the 4 Methods in Subsection 4.1 when the optimal policy is demonstrated and a suboptimal policy is demonstrated. We compared the performance of an agent using the information provided by the demonstration with that of an agent that has no previous information on the task. In all situations we allowed both agents to learn for 200 time steps using an ε-greedy policy with decaying ε.

Table 1 provides the percentage of time (out of the 200 time steps) that the agents are able to reach the goal state (empty table). For the sake of comparison, we also provide the performance of a "pure" reinforcement learner.

Table 1. Results obtained with Methods 4.1.1 through 4.1.4 using optimal and suboptimal demonstrations.

	Optimal	**Suboptimal**
Pure RL	34.6 %	32.4 %
Method 4.1.1	41.5 %	40.5 %
Method 4.1.2	41.5 %	37.5 %
Method 4.1.3	41.5 %	39.0 %
Method 4.1.4	42.0 %	41.0 %

From Table 1 it is evident that the performance of the learning algorithm is improved when considering a demonstration, since the agents were able to reach the goal state (and thus complete the task) more often. To have a clearer understanding of how this translates in terms of the learning process, we present in Figures 3 through 6 the total reward obtained during learning.

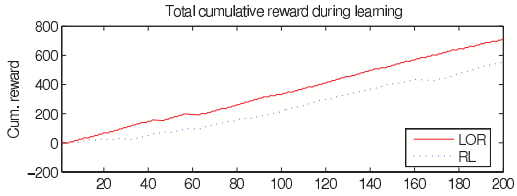


Figure 3. Total reward obtained with Method 4.1.1 over the time-frame of 200 steps and corresponding exploration probabilities when the demonstrator follows an optimal policy.

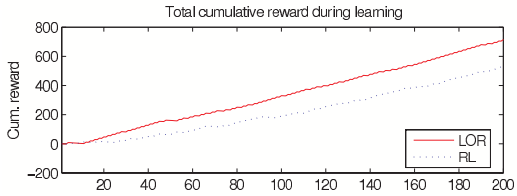


Figure 4. Total reward obtained with Method 4.1.2 over the time-frame of 200 steps and corresponding exploration probabilities when the demonstrator follows an optimal policy.

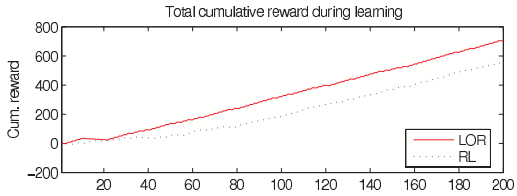


Figure 5. Total reward obtained with Method 4.1.3 over the time-frame of 200 steps and corresponding exploration probabilities when the demonstrator follows an optimal policy.

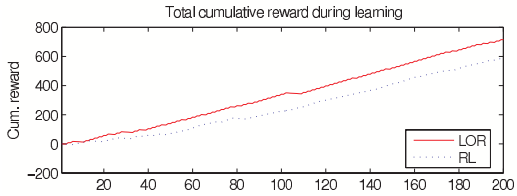


Figure 6. Total reward obtained with Method 4.1.4 over the time-frame of 200 steps and corresponding exploration probabilities when the demonstrator follows an optimal policy.

In the plots, the slope of the performance curve indicates how good is the learnt policy. It is clear that, in all methods, the provided information gives the learning agent a significative advantage: in the beginning of the learning process, the "greedy" action for the agents that were provided a demonstration is much more informed than that of the pure RL learner. This means that the demonstration provides the learner with a *knowledge boost* by improving the estimative of the optimal Q-function and thus speeding up the learning.

Notice that, in all these methods, the demonstration provides only informed initial estimates for Q^*, thus improving the initial performance of the agent. However, since this initial estimate is then properly adjusted by the learning algorithm, the sub-optimality of the demonstrated policy does not affect the performance of the learner.

In a second set of experiments we tested Method 4.2.1 from Subsection 4.2. To evaluate the performance of the method, we explicitly observed the learnt policy when the demonstrated policy is optimal and when it is not. The results are summarized in Table 2. We denoted by 0 the empty slot, by B the large ball, by c the cube and by b the small ball.

Notice that both learnt strategies are optimal. This is due to the fact

Table 2. Learnt policies with Method 4.2.1 using optimal and suboptimal demonstrations.

	Optimal	**Suboptimal**
$(0,0)$	TL	TL
$(0,B)$	TR	TR
$(0,c)$	GR	GR
$(0,b)$	KR	KR
$(B,0)$	TL	TL
(B,B)	TR	TL
(B,c)	TL	TL
(B,b)	TL	TL
$(c,0)$	GL	GL
(c,B)	TR	TR
(c,c)	GR	GR
(c,b)	GL	GL
$(b,0)$	KL	KL
(b,B)	TR	TR
(b,c)	GR	KL
(b,b)	KL	KL

that, in considering the same final state, the reward function obtained by Method 4.2.1 is the same independently of the actual policy used to demonstrate. And, in this particular case, it matches exactly the reward function considered in the previous examples, thus giving rise to the same policy.

Finally, we tested Method 4.2.2 from Subsection 4.2. As in the previous experiment, we evaluate the performance of the method by explicitly observing the learnt policy when the demonstrated policy is optimal and when it is not. The results are summarized in Table 3. In the third column we also present the results obtained with Method 4.2.2 using an optimal policy, but where the model is also estimated from the demonstration. The table elements in bold denote "suboptimal" actions.

Table 3. Learnt policies with Method 4.2.2 using optimal and suboptimal demonstrations.

	Optimal	**Suboptimal**	**No Model**
$(0,0)$	TL	TL	TL
$(0,B)$	TR	**TL**	**TL**
$(0,c)$	GR	**TR**	GR
$(0,b)$	KR	KR	KR
$(B,0)$	TL	**KL**	TL
(B,B)	TR	**GL**	TL
(B,c)	TL	**TR**	TL
(B,b)	TL	**TR**	**TR**
$(c,0)$	GL	**TL**	GL
(c,B)	TR	**GL**	**TL**
(c,c)	GR	GR	**TR**
(c,b)	GL	**KR**	**KL**
$(b,0)$	KL	KL	**TR**
(b,B)	TR	**KL**	**KL**
(b,c)	GR	**TL**	**TL**
(b,b)	KL	**TR**	**TR**

We emphasize the policy obtained with Method 4.2.2 when the demonstrated policy is suboptimal (and the agent has little confidence on the observed policy). Recall that this method determines a likely reward function for which demonstrated policy, we expect the performance of this method to be affected by the sub-optimality of the demonstrated policy. Notice that the policy learnt from a suboptimal demonstration is even worse than that learnt in the absence of a model with an optimal demonstration (third column of Table 3).

To conclude this section, we present the images obtained by experimenting Method 4.2.1 in a real robot. The robot is capable of recognizing the actions Grasp, Touch and Kick as well as the objects on the table (to details refer to [7]). Figure 7 presents the robot following the task it learned after having observed it.

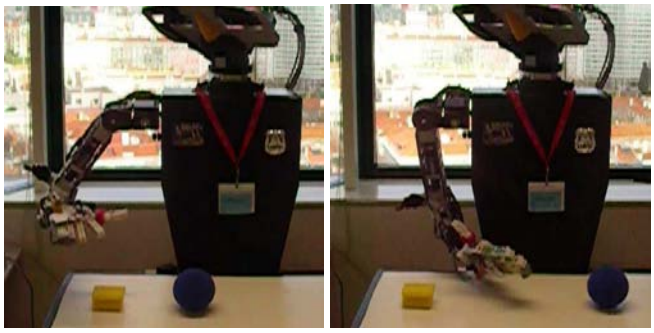


Figure 7. Robot following the learned task.

6 CONCLUSIONS

In this paper, we proposed an unified formalism to address imitation learning and RL problems. Using this formalism, we analyzed several imitation-like learning mechanisms, such as stimulus enhancement, response facilitation, contextual learning and Emulation. These mechanisms can lead to imitative behavior without being imitation in the stricter sense of the concept. In this formalism, which we refer as the *learning by observation and reward* (LOR), these behaviors can be summarized as:

- *Stimulus enhancement*: biases exploration toward the observed partial policy;
- *Contextual learning*: uses the observed transitions to improve the model of the world;
- *Response facilitation*: uses the observed rewards to accelerate learning;
- *Emulation*: selects a final state, defines a reward and learns using standard RL methods.

As one of our major contributions was to unify all of these mechanisms in the same framework. We demonstrated that this modulation is possible and the resulting behavior of the learner matches the descriptions of the behavior in animals. We saw that when learning from others there are many sources of information and each of them can be exploited individual or in combination.

The results presented clearly established one of the known advantages of imitation learning: the imitation learner acquired the optimal policy for the problem faster than a learner following a standard trial-and-error learning strategy. We emphasize that, in most cases above, the agent would be able to learn the task on its own—the learner did not extract the solution from the demonstration. Instead, the demonstration provided *tips* to help solve the task that the learner used to discover how to complete the task more efficiently.

It is interesting to note that, as these mechanisms do not rely completely on the details of the demonstration, they can also learn the optimal policy even when the demonstration was sub-optimal. The learner can thus look at someone performing a task and then understand the goal of the task and outperform the teacher.

We also emphasize the difference between imitation-like methods and "pure" imitation methods. In a pure imitation system, the found solution should not exist in the learner repertoire; or it should not be possible to know the task if were not for the demonstration. In our model we assumed that, without the demonstration, the agent does

not know the task (there is no reward mechanism). In imitation-like methods, the learner can always learn the task on its own.

In our proposed LOR framework it is not easy to distinguish between action-level and program-level learning, since the important steps of the demonstration are abstract concepts that can be implemented in several ways. We intended to address this problem with further detail by defining an hierarchical learner where we can define actions at several "resolutions".

Also, the demonstrations used throughout the paper do not illustrate the full richness of the expected behavior of the different methods, mainly due to the great simplicity of the task—the state and action spaces are small and there is a unique optimal solution. Finally, as already mentioned, we also intend to study the effects of partial observability of state and action.

Acknowledgements

This work was partially supported by Programa Operacional Sociedade do Conhecimento (POS_C) that includes FEDER funds, and by the EU-Project Robotcub. The first author acknowledges the PhD grant SFRH/BD/3074/2000.

References

[1] Richard Byrne. *The Thinking Ape Evolutionary Origins of Intelligence*. Oxford University Press, 1995.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[3] S. Schaal. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[4] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Phil. Trans. of the Royal Society of London: Series B, Biological Sciences*, 358(1431):537–547, 2003.

[5] Manuel Lopes and José Santos-Victor. A developmental roadmap for task learning by imitation in robots. *IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics*, 37(2), 2007.

[6] Hideki Kozima, Cocoro Nakagawa, and Hiroyuki Yano. Emergence of imitation mediated by objects. In *2nd Int. Workshop on Epigenetic Robotics*, 2002.

[7] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. Learning affordances objects: From sensory motor maps to imitation. Technical report, Instituto de Sistemas e Robótica, Lisbon, Portugal, Feb 2007.

[8] Chrystioher L. Nehaniv and Kerstin Dautenhahn. Like me? - measures of correspondence and imitation. *Cybernetics and Systems*, 32:11–51, 2001.

[9] Manuel Lopes and José Santos-Victor. Visual transformations in gesture imitation: What you see is what you do. In *IEEE Int. Conf. Robotics and Automation*, 2003.

[10] R. Zöllner and R. Dillmann. Using multiple probabilistic hypothesis for programming one and two hand manipulation by demonstration. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2003.

[11] H. Bekkering, A. Wohlschläger, and M. Gattis. Imitation of gestures in children is goal-directed. *Quarterly J. Experimental Psychology*, 53A:153–164, 2000.

[12] György Gergely, Harold Bekkering, and Ildikó Király. Rational imitation in preverbal infants. *Nature*, 415:755, 2002.

[13] Richard W. Byrne. Imitation of novel complex actions: What does the evidence from animals mean? *Advances in the Study of Bahaviour*, 31:77–105, 2002.

[14] R. W. Byrne and A.E. Russon. Learning by imitation: a hierarchical approach. *Behav Brain Sci*, pages 667–84, 1998.

[15] J.S. Bruner. Nature and use of immaturity. *American Psychologist*, 27:687–708, 1972.

[16] Minoru Asada, Yuichiro Yoshikawa, and Koh Hosoda. Learning by observation without three-dimensional reconstruction. In *Intelligent Autonomous Systems (IAS-6)*, 2000.

[17] C. Nehaniv and K. Dautenhahn. Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation. In *European Workshop on Learning Robots*, 1998.

[18] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances,. In *ICRA*, Taipei, Taiwan, 2003.

[19] Aude Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47:2-3, 2004.

[20] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[21] Christopher J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, University of Cambridge, May 1989.

[22] Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit imitation. *J. Artificial Intelligence Research*, 19:569–629, 2003.

[23] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. Proc. 20th Int. Joint Conf. Artificial Intelligence, 2007. (to appear).

[24] M.T. Rosenstein and A.G. Barto. Supervised actor-critic reinforcement learning. In *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*. John Wiley & Sons, Inc., 2004.

[25] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *14th International Conference on Machine Learning*, pages 12–20. Morgan Kaufmann, 1997.

[26] Andrew Y. Ng and Stuart J. Russel. Algorithms for inverse reinforcement learning. In *Proc. 17th Int. Conf. Machine Learning*, pages 663–670, 2000.

[27] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. 21st Int. Conf. Machine Learning*, pages 1–8, 2004.

When Training Engenders Failure to Imitate in Grey Parrots (*Psittacus erithacus*)

Irene M. Pepperberg[1]

Abstract. The initial study on avian behaviour [1] was not designed to examine imitation, but nevertheless provided information concerning issues involving imitation. Four Grey parrots (*Psittacus erithacus*) were tested on their ability to obtain an item suspended from a string such that multiple, repeated, coordinated beak-foot actions were required for success (e.g., [2]). Those birds with little training to use referential English requests (e.g., "I want X") succeeded, whereas birds who could vocally request the suspended item failed to obtain the object themselves and instead engaged in repeated requesting [1]. Interestingly, even after subsequent, multiple observations of the actions of a successful parrot, the unsuccessful birds persevered in vocal requests or ignored the task, possibly retreating into learned helplessness. Such data emphasize three points: First, the entire behavioural repertoire and history must be examined in studies that try to determine whether animals act intelligently and/or can imitate; second, parrots can attempt to direct humans to assist them in achieving their goals, and such behaviour—although clearly complex—might lead them to fail certain tasks designed to test intelligence; third, even for a species known for imitative behaviour (physical as well as vocal [3]), imitation may not be expressed if it must overcome previous training.

1 INTRODUCTION

Defining and evaluating intelligence is a dauntless task with respect to humans (e.g., [4]) and is even more so with respect to nonhumans [5]: To examine nonhuman abilities, should an experimenter administer what are basically human tasks to nonhumans, making minimal concessions and adaptations to, for example, take into account their tendencies to peck a lit button rather than a computer keyboard, or instead restructure the tasks to accommodate any significantly different species-specific traits, such as poor vision and excellent olfaction? No simple solution exists, but one possible route through these difficulties is to examine not the ability to solve a specific problem but rather the *processes* whereby problems of ecological or ethological interest are solved. Consequently, researchers have become enamored of two types of studies—those involving insight and imitation. The first is favored because success suggests that the subject has formed a sophisticated representation of the problem and attained a solution via mental rather than physical trial-and-error, implying such an advanced understanding of—and memory for—actions and outcomes that physical experimentation is unnecessary. The second has become popular because success suggests that the subject can view, conceptualize, and then recreate from his/her own perspective, novel and improbable actions that lead to successful solution of a novel problem [6], also implying advanced cognitive processing skills. (The question also arises as to whether emulation—the attainment of the demonstrated goal via any means (e.g., [7])—is more or less advanced than imitation, but that is a separate issue). Of course, unless the experimenter knows the complete history of the subject, success or failure on a task might not be an accurate evaluation of capacities for insight or imitation, but rather relate to prior experience that may have either potentiated or blocked the targeted behaviour. And therein lies the question to be addressed in processing the results of both the prior [1] and present studies.

The initial experiment [1] was designed to examine whether Grey parrots (*Psittacus erithacus*) were capable of insightful behaviour; only later were the birds tested on their imitative competence. The task chosen, to obtain a special food treat suspended by a string, by reaching down, pulling up a loop of string onto the perch, stepping on the loop to secure it, and repeating the sequence several times (e.g., to demonstrate an understanding of intentional means-end behaviour; see review in [8]) has been previously used to assess "insight" in several avian species; simply reaching down for the food is not sufficient [2,9]. Not all birds succeed on this task [2,6,10; for more recent studies and reviews of older studies, see 11,12], suggesting that the necessary action pattern requires a higher-order cognitive ability that is prevalent neither among species nor within a given species.

Clearly, the extent to which the task is solved individually via insight might be affected by prior physical manipulative experience [11], but could prior training affect the ability to derive a solution via imitation of an expert? I had previously found [1] that for Grey parrots the capacity (or possibly willingness) to use insight could be tempered by a nonphysical type of training: specifically, that of my birds having learned to demand access to various objects vocally. Precisely because some of my birds can routinely request items from a human, without the need to work to obtain it on their own, two of the four birds tested (those with this vocal ability) failed the test of insight, persisting in their vocal requests. Possibly, a bird that responds with repeated requests, although ostensibly failing at the given task, could be considered to have demonstrated instead an alternative higher-order intelligence, in that it knows how to manipulate another individual to access its wants. Might, however, this ability to manipulate others interfere not only with the use of insight but also with the use of imitation? Two birds in the prior study observed but did not imitate after viewing a single trial by a successful demonstrator; the present study was performed to determine if repeated viewings of a demonstrator might be required to initiate any observational learning of a physical act.

I will review the initial study (reported in [1]), then describe subsequent trials to determine whether the parrots would engage in any behaviour related to observational learning of the string-pulling task. I compare my findings to

[1] Departments of Psychology, Harvard and Brandeis Universities, Cambridge and Waltham, MA, USA.

those using the same task for other avian species [11,12].

2 METHOD

2.1 Subjects

Four Grey parrots were the subjects of the insight part of the study [1] and three of the four were also involved in the imitation part of the study. Kyaaro, obtained from a breeder at 3 months old, was, when tested on insight in 1995, 4¾ years old and had had about four years of training on interspecies communication; much of his instruction had, however, involved unsuccessful video and audio exposure and his vocabulary was limited to a few object labels (i.e., parrots do not learn referential labels if training is via video or audiotapes [13,14,15]). He was removed from the laboratory in 2001, and did not participate in further experiments, such as the imitation study. At the time of most of the trials, in 2003, Alex was 27 years old and had been the subject of experiments on avian cognition and interspecies communication for 26 years [16]; his training involved human modeling [17] and his vocabulary included labels for over 50 different objects, seven colors, five shapes, numbers up to 6, three categories, and many functional phrases (e.g., "I want X", "Wanna go Y", "Come here", "Go pick up X", etc.). He had had one insight trial in 1995. In 2003, Griffin was 8 years old; he had been obtained from a breeder when 7½ weeks old and had been the subject of studies similar to those involving Alex; his unsuccessful video training experiments, unlike those of Kyaaro, had been limited to only a few labels (e.g., [15]). His vocabulary, although not as extensive as Alex's, therefore contained most of the same commands and functional phrases. He had also been the subject of a study on the simultaneous development of object and label combinations [18]. Arthur (aka Wart) was about 4½ years old in 2003; he had been obtained from his hand-feeder when he was about a year old, but most of his training had involved studies on animal-human computer interfaces [19], and his vocalizations were limited to just a few labels [14]; he could state "want some" when a trainer had something he desired, but could not specify an item for trainers to retrieve. Alex, Griffin, and Arthur participated in one imitation trial in 2003 and in additional imitation trials in 2006.

2.2 Apparatus

As reported in [1], birds were tested on parrot "T" stands. For Alex, Griffin, and Arthur, almonds (a favorite food) or pieces of blackboard chalk (a less favored item, but one with which they had interacted in the past) were suspended at the end of 60 cm long chains of plastic links hung from the end of a "T" stand; red or green oval links were ~ 2.5 cm long and 1.2 cm wide, blue triangular links were ~ 2.5 cm long and 2.5 cm at their widest. These chains would provide the birds with adequate purchase if they attempted to obtain the suspended items. For Kyaaro, a favorite bell was suspended at one end of the stand from a silken cord approximately 0.6 cm in thickness and about 60 cm in length; he could not chew through such cords and did not exhibit any fear of them but was afraid of the plastic chains we used for the other birds.

2.3 Procedure

Again as reported in [1], birds were initially examined individually on the test for insight. Each bird was placed on the "T" stand after a targeted item was suspended; trainers then pointed to the object. If a bird did not seem interested at first, it was told to "Pick up the nut/bell/chalk/treat". (All our birds respond to such commands if there is a single choice on, e.g., a tray; given multiple choices, they take their favorite item [16].) Birds were given several minutes in which to attempt the procedure; if they did not succeed, make an attempt, or demonstrate interest within 5 min, the trial was ended. Each bird except Alex was given three trials in its first session; Alex had only one trial in 1995. Two weeks after the first 2003 trials, Arthur was given a simultaneous choice between a nut hung from one chain (red, oval links) and chalk from another (blue, triangular links). Two months later (the delay was to avoid the possibility of training or massed-trial learning) Alex, Griffin, and Arthur were then given three more trials involving a single chain, with the less desirable item (a piece of chalk) suspended in the first two of the three trials, and a nut in the third. The intention was to see whether the type of reward affected their behaviour and if they could spontaneously solve the problem, not if they could learn to obtain their treat.

Note that none of the birds had received any training on this task prior to testing. Any toys hanging in their cages were suspended on short metal chains (at most, 7.5 cm long) such that each toy was at approximately beak level when birds were perched; thus they would not have been able to practice the maneuver. Only Arthur had had a toy suspended from a perch by a long chain (~30 cm) prior to testing, and it was his demonstration of the targeted behaviour as soon as the toy had been suspended that prompted formal testing.

After Alex and Griffin failed and Arthur succeeded on this test of insight (see **RESULTS**, full details are in [1]), the former two birds were allowed to watch Arthur once in 2003 and six times (twice each day for three non-consecutive days) in 2006. Each time, an almond was suspended from a plastic oval link chain at the end of Arthur's "T" stand as during the insight trials. Alex and Griffin were placed on their own "T" stands, less than 1 m from Arthur's stand (out of reach but in clear view), and then Arthur was placed on his stand and allowed to retrieve the nut. Alex's and Griffin's interest in the nut ensured that they observed Arthur's actions.

3 RESULTS

3.1 Insight trials

3.1.1 Kyaaro and Arthur

On their first exposures [1], both Arthur and Kyaaro immediately performed the targeted action of pulling, stepping, and repeating the behaviour so as to obtain the desired treat; they repeated their actions correctly each time without any hesitation for a total of three trials. For both

birds, the actions were not necessarily performed smoothly (occasionally they had to make more than one attempt at grasping the chain or cord that sometimes began to swing as the trial progressed), but they acted consistently and with perseverance. (See video S1 in [1] for one of Arthur's trials.) Kyaaro had no more trials.

On the choice trials ([1], between nut and chalk), Arthur first performed the series of manipulations to obtain and eat the nut (i.e., chose the chain with the nut first), then repeated the manipulations with the chain holding the chalk. He dropped the chalk immediately after obtaining it.

On the final set of three trials [1], Arthur successfully performed the operations to obtain the chalk both times; he quickly discarded the chalk after extracting it from the chain. He also succeeded with the nut, which he dropped, seemingly by accident.

3.1.2 Alex and Griffin

On their first trials [1], neither Alex nor Griffin made any attempts at recovering the nuts. In Alex's only trial in 1995 and in his subsequent trials in 2003, he, like Griffin in 2003, looked at the nuts, looked at the trainer, and said "Want nut". To the trainer's command "Go pick up nut", they both replied "Want nut"; this verbal interplay was repeated several times during each trial. (See video S2 in [1] for part of one of Alex's trials.) In Alex's case, the volume and intensity of the request increased in one trial with the trainer's failure to comply.

In their final three trials [1], Alex and Griffin both completely ignored the chalk and, interestingly, then also ignored the nut; that is, they made no requests for either object nor did they engage in any action required to obtain either object.

3.2 Imitation trials

After Alex's and Griffin's first failure in 2003, they observed one of Arthur's successful trials, but their behaviour did not change; that is, they consistently requested the nut from the trainer and failed to make any attempts themselves [1]. The single-trial session was ended for both Alex and Griffin without their having succeeded.

In 2006, both birds were again given opportunities to observe Arthur's successful trials; they again failed to engage in any form of observational learning. On the first two trials (session one), they watched and requested the nuts vocally; on the next two trials (session two, held two days later), they watched but Alex did not make requests while Griffin continued to do so, and on the final two trials (session three, held about four weeks later), they again watched and both requested the nuts.

4 DISCUSSION

The results of these studies have implications for evaluating the effects of prior training on both insight and imitation. Detailed discussion of how training affects insightful behavior can be found in the original article [1], which I will review only briefly. I will concentrate instead on the effects of previous training on imitation and compare the results with data on other avian species.

In terms of insight, the noteworthy result of the prior experiment [1] was that the two parrots with limited vocabularies immediately acted out the correct physical tasks to obtain their treats, whereas the parrots that had received considerably more effective training in referential English speech attempted instead to manipulate their trainer. These birds' requesting behaviour appeared intentional: They were asking that trainers do something for them, in very specific, fairly stress-free circumstances and in a very direct manner [1]. They were not treating humans as a physical object to be used (e.g., as a stepping-stone to reach something desired; see [20]), but were engaging in deliberate communication as a problem-solving strategy, which is a fairly advanced stage of development, even for human infants (see [20,21]). These birds acted just as they do when they want other treats that are not within their reach (e.g., [16]), thereby cross-applying (transferring) behaviour patterns learned in one situation to another, which is also considered a hallmark of intelligent behaviour [22]. According to some researchers (e.g., [23]), the adaptive value of using a referential communication code to benefit oneself at the expense of others is viewed as an advanced, essentially human trait.

Why Alex and Griffin did not continue to request the suspended nuts in their final trials in 2003 nor Alex in his middle imitation trials in 2006 was not clear. They possibly had learned in previous trials that no trainer would assist them and that requesting the nut was useless (a form of learned helplessness [24]); other reasons for their lack of action (including string-pulling) are discussed in detail in [1].

Here, however, I wish to focus on the birds' lack of physical imitative ability. Why were Griffin and Alex unable—or at least unwilling—to reproduce an observed behaviour to acquire a desired treat? Several issues are of note.

First, my Grey parrots have shown, over the course of almost 30 years of study, a facility for accurately reproducing English speech. Moreover, these referential vocal abilities all derive from a social learning paradigm [16,17], thereby demonstrating the parrots' competence for observational learning. Although they use a different vocal apparatus than humans to produce speech, in many cases their articulatory acts would indeed seem to qualify as imitation [1,25,26,27]; such data suggest that some form of imitation is within the purview of the Grey parrot.

Second, a Grey parrot in different laboratory has been shown to reproduce human physical actions, such as waving a foot after seeing a human wave his hand [3]. Although the extent to which such behaviour patterns are novel and would fit Thorpe's definition of imitation [6] is unclear, the capacity of the bird in question to integrate observed physical actions into its behavioural repertoire suggests that this ability is also within the purview of the Grey parrot.

Third, in species such as goldfinches (*Carduelis carduelis*) and siskins (*Carduelis spinus*), not only do only a percentage of tested birds (23% of 52 the former, 62% of 29 of the latter) solve the string-pulling problem, but only another small percentage (25% of the former species, 10%

of the latter) who fail by themselves achieve any form of success after observing successful birds [11]. Too, those who achieve success via observation often did so by emulation—achieving the goal by a different method—rather than by imitating the actions of the demonstrators [11]. Most of those birds that consistently failed, even after being exposed to a demonstrator, did not fail because of lack of observational experience [11]. Assuming that such behaviour can be extrapolated to parrots—a likely supposition given the work of Huber and his colleagues [12,28,29], which demonstrated considerable individual differences and various levels of imitation and emulation in keas—birds (including parrots) likely exhibit individual differences in their ability or motivation to reproduce observed actions.

Given that Grey parrots have demonstrated competence in what appear to be related tasks of observational learning, I suggest that, whatever individual differences might exist between Alex, Griffin, and Arthur, that Alex's and Griffin's failure to reproduce Arthur's actions in the string-pulling task was a consequence of their previous training that emphasized the vocal mode and a paradigm in which humans would diligently respond to their vocal requests. Such training may have reduced their motivation to act (physically) on their own. Granted, neither Alex nor Griffin had had significant experience in the kind of physical manipulations (e.g., pulling at branches or twigs to obtain food) that might not only engender string-pulling but might also potentiate imitation of related physical actions [11], but neither had Kyaaro nor Arthur had such experience, and all birds had been given numerous objects that they could chew or tear apart, pick up or toss with foot or beak. Interestingly, Alex and Griffin, in contrast to Kyaaro and Arthur, were given tasks in which covers needed to be removed to expose hidden objects (e.g., [30] and references therein); Griffin also had demonstrated some proficiency in combining objects [18]. Note, however, that all actions were done with their beaks. Possibly, as was suggested in [1], for Alex and Griffin, successful vocal training may have caused communication (or at least beak-related) areas in the brain to develop to the detriment of those used to control complex, sequential physical actions involving both limbs and beaks (Heinrich, pers. comm.). The string-pulling task involves eye-foot-beak coordination and thus may have required brain areas in addition to those involved in solely beak-driven combinations such as stacking or removing cups and vocalizing. If true, this explanation does not detract from the complexity of the vocal behaviour, but rather provides a rationale for the Alex's and Griffin's vocal rather than physical actions.

Another issue might be the dominance hierarchy of the birds in the laboratory—would Alex and Griffin be willing to reproduce the behaviour of an individual in a position clearly subordinate to theirs? Arthur is the youngest, most recent addition to the lab, and by default the lowest ranking bird. One might imagine that having humans demonstrate the targeted string-pulling behaviour pattern, as they do with vocal patterns, might be preferable, but that option (hand-over-hand, or even an attempted mouth-hand demonstration) would not allow the birds to see how *they* might perform the task and could even be viewed by the birds as an acquiescence to their demands, not as a demonstration. (One such human demonstration, performed just before the writing of this manuscript, engendered the not-unexpected request for the retrieved nut.) Arguably, Alex's and Griffin's demands that the trainers do the task might be taken as evidence that they consider themselves dominant to the humans in the laboratory; clearly, humans do spend as much time acceding to their demands as querying and thus making demands of them. Consistent with such a view is the possibility that Arthur, subordinate to the other birds, might also be seen as subordinate to humans because he cannot ask trainers to carry out his demands and, thus, was unworthy of imitating.

I suspect that, in order to demonstrate that Alex and Griffin could engage in either insightful behaviour or a form of imitation that involves object manipulation, I would have to devise a task that would be intriguing and motivating enough to spontaneously override their prior training. For obvious reasons (continuing experiments on vocal learning and cognitive processing), extinguishing their previous training is not an option, and the parrots' overt distress upon the exit of trainers [1] precludes at present carrying out the study (Bugnyar, pers. comm.) involving videotaping Alex and Griffin in the absence of human observers.

5 CONCLUSION

In sum, two parrots that had limited use of vocal requests exhibited behaviour similar to the insightful food retrieval displays of, for example, Heinrich's ravens [2] and Funk's kakarikis [9]; the two parrots who could make specific vocal requests did so instead, and continued to do so even after observing the successful retrieval by another parrot. Such data emphasize three points. First, that the entire behavioural repertoire and history must be examined in studies that try to determine whether animals act insightfully or are capable of imitation; second, that parrots can attempt to direct humans to assist them in obtaining their goals; and third, that such behaviour—although clearly complex—might lead them to fail certain tasks.

ACKNOWLEDGMENTS

I thank the many donors to *The Alex Foundation* for their support of this research, particularly the Pearl Family Foundation, Janice Boyd, Kathryn and Walter McAdams, Janet Trumbule, John Paton, Nancy Clark and Bill Broach, Megumi Oka and the Makioka Foundation, Michael and Deborah Smith, Alex and Michael Shuman, Will Lipton, Greg LaMorte, Nancy Chambers, Carl and Leigh Ann Hartsfield, and Deborah and Robert Goodale.

REFERENCES

[1] I.M. Pepperberg, '"Insightful" string-pulling in Grey parrots (*Psittacus erithacus*) is affected by vocal competence', *Animal Cognition*, **6**, 263-266, (2004).

[2] B. Heinrich, 'An experimental investigation of insight in Common Ravens (*Corvus corax*)', *Auk*, **112**, 994-1003, (1995).

[3] B. Moore, 'Avian movement imitation and a new form of

mimicry: Tracing the evolution of a complex form of learning', *Behaviour*, **122**, 231-263, (1992).
[4] H. Gardner, *Multiple Intelligences: New Horizons.* Perseus Group, Basic Books, New York, 2006.
[5] I.M. Pepperberg, Evolution of avian intelligence. In *The Evolution of Intelligence*, R. Sternberg and J. Kaufman, eds., Erlbaum, Mahwah, NJ, 2001.
[6] W.H. Thorpe, *Learning and Instinct in Animals*, 2nd Ed. Harvard University Press, Cambridge, MA, 1963.
[7] M. Tomasello, *The Cultural Origins of Human Cognition*, Harvard University Press Cambridge, MA, 1999.
[8] P. Willatts, 'Development of means-end behavior in young infants: Pulling a support to retrieve a distant object', *Developmental Psychology*, **35**, 651-667, (1999).
[9] M.S. Funk, 'Problem solving skills in young yellow-crowned parakeets (*Cyanoramphus auriceps*)', *Animal Cognition*, **5**, 167-176, (2002).
[10] M.Vince, 'String pulling in birds. III. The successful response in greenfinches and canaries', *Behaviour*, **17**, 103-129, (1961).
[11] U. Seibt and W. Wickler, 'Individuality in problem solving: String pulling in two *Carduelis* species (Aves: Passerformes)', *Ethology*, **112**, 493-502, (2006).
[12] D. Werdenich and L. Huber, 'A case of quick problem solving in birds: String pulling in keas, *Nestor notabilis*', *Animal Behaviour*, **71**, 855-863, (2006).
[13] I.M. Pepperberg, 'Vocal learning in African Grey parrots: effects of social interaction', *Auk*, **111**, 300-313 (1994).
[14] I.M. Pepperberg and S.R. Wilkes, 'Lack of referential vocal learning from LCD video by Grey Parrots *(Psittacus erithacus)*', *Interaction Studies*, **5**, 75-97 (2004).
[15] I.M. Pepperberg, L.I. Gardiner, and L.J. Luttrell, 'Limited contextual vocal learning in the Grey parrot (*Psittacus erithacus*): the effect of co-viewers on videotaped instruction', *Journal of Comparative Psychology*, **113**, 158-172, (1999).
[16] I.M. Pepperberg, *The Alex Studies*, Harvard University Press, Cambridge, MA, 1999.
[17] I.M. Pepperberg, 'Functional vocalizations by an African Grey parrot (*Psittacus erithacus*)', *Zeitshrift für Tierpsychologie*, **55**, 139-160, (1981).
[18] I.M. Pepperberg and H. Shive, 'Simultaneous development of vocal and physical object combinations by a Grey Parrot (*Psittacus erithacus*): Bottle caps, lids, and labels', *Journal of Comparative Psychology*, **115**, 376-384, (2001).
[19] I.M. Pepperberg, *The Wired Kingdom*. Conference at the MIT Media Lab, April. 2000.
[20] J.C. Gómez, The emergence of intentional communication as a problem-solving strategy in the gorilla. In *"Language" and Intelligence in Monkeys and Apes: Comparative Developmental Perspectives*, S.T. Parker and K.R. Gibson eds., Cambridge University Press, New York, 1990.
[21] R. Case, *Intellectual Development: Birth to Adulthood.* Academic Press, Orlando, FL, 1984.
[22] P. Rozin, The evolution of intelligence and access to the cognitive unconscious. In *Progress in Psychobiology and Physiological Psychology*, (Vol. 6), J.M. Sprague and A.N. Epstein, eds., Academic Press, NY, 1976.
[23] D. Kemmerer, 'What about the increasing adaptive value of manipulative language use?', *Behavioral & Brain Sciences*, **19**, 546-548, (1996).
[24] M.E. Seligman and S.F. Maier, 'Failure to escape traumatic shock', *Journal of Experimental Psychology*, **74**, 1-9, (1967).
[25] D.K. Patterson and I.M. Pepperberg, 'A comparative study of human and Grey parrot phonation: Acoustic and articulatory correlates of stop consonants', *Journal of the Acoustical Society of America*, **103**, 2197-2213, (1998).
[26] D.K. Warren, D.K. Patterson, and I.M. Pepperberg, 'Mechanisms of American English vowel production in a Grey Parrot (*Psittacus erithacus*)', *Auk*, **113**, 41-58, (1996).
[27] I.M. Pepperberg, Training behavior by imitation: from parrots to people…to robots. In *Proceedings of the AISB '03 Second International Symposium on Imitation in Animals and Artifacts*, K. Dautenhahn and C. Nehaniv, eds., University of Wales, 2003.
[28] L. Huber, S. Rechberger, and M. Taborsky, 'Social learning affects object exploration and manipulation in keas, *Nestor notabilis*', *Animal Behaviour*, **62**, 945-954, (2001).
[29] G. Gajdon, N. Fijn, and L. Huber, 'Testing social learning in a wild mountain parrot, the kea (*Nestor notabilis*)', *Learning & Behavior*, **32**, 62-71. (2004).
[30] I.M. Pepperberg, M.R. Willner, and L.B. Gravitz, 'Development of Piagetian object permanence in a Grey parrot (*Psittacus erithacus*)', *Journal of Comparative Psychology*, **111**, 63-75, (1997).

Imitative learning in monkeys

Ludwig Huber[1], Bernhard Voelkl[1] and Friederike Range[1]

Abstract[1]. Imitative learning has received high levels of attention due to its supposed role in the development of culture, language and self-identification and the cognitive demands it poses on the individual. Although monkeys possess mirror neurons, show neonatal imitation, recognize when being imitated and copy an expert's use of a rule, their capacity of action imitation has been doubted by most imitation researchers so far. Here I will argue that imitation in the original definition of learning to do an act from seeing it done must be distinguished from other forms of "copying", in which the content of the copy is not the behavior of the model but the result of the demonstrated action, its goal or the intention of the demonstrator. Then I will describe several experiments with captive common marmosets (*Callithrix jacchus*) that show that these monkeys can use the same overall pattern of a technique to open a food box, or the same body part as the model, or – above all – can precisely copy the movement pattern of an action that a skilful model has demonstrated. On the basis of this cumulative evidence of imitation in non-human primates I will question the frequently expressed notion that imitation is a relatively recent invention in the hominoid lineage and will discuss its implications for the currently available theories of the underlying neuronal mechanism.

1 MONKEY SEE, MONKEY DO

According to Byrne [1], imitation research has focused on one of two distinct problems. The one favored by cognitive neuroscientists is the 'correspondence' problem, asking how is it possible for actions as seen to be matched with actions as imitated? The other, favored by ethologists and comparative psychologists, is the 'transfer of skill' problem, asking how is it possible for novel, complex behaviors to be acquired by observation? Despite various approaches to, and definitions of, imitation [2-5], most scholars agree that when an individual replicates an action that it has observed being performed by another individual it requires a matching system that allows conversion of observed actions by others into actions executed by oneself. In other words, visual input needs to be transformed into corresponding motor output. However, most currently available models of imitation require that the observers had possessed a motor representation of the demonstrated action already before they observe it being performed by the model. But how can new skills be acquired if the essence of imitation lies in the activation or facilitation of responses already in the repertoire of the observer? Imitative learning in the sense of the acquisition of new skills by observation must therefore be distinguished from response facilitation [6], priming, stimulus enhancement and other forms perception-motor coupling, let alone many forms of social influences [7-10].

A common conclusion about social learning among primates was that apes imitate in various forms [11], but that monkeys, despite a century's efforts, had not been shown to imitate [7, 12-14]. Although the sweet potato washing of Japanese macaques on Koshima Islet is perhaps the best known and most frequently cited example of the formation of traditions in nonhuman animals [15, 16], it has been questioned whether social learning, let alone imitation, is really involved [17]. Furthermore, Visalberghi and Fragaszy have made several attempts to find out whether Capuchin monkeys learn by observation of a skilful model how to use an object as a tool [13, 14]. However, all these attempts failed.

Recently, the picture of the monkeys' failure to imitate has been seriously doubted, because macaques show cognitive imitation by copying an expert's use of a rule [18], recognize when they are being imitated [19] and imitate adult facial movements as neonates [20]. Also, the discovery that rhesus monkeys have "mirror neurons"—neurons that fire both when monkeys watch another animal perform a goal-directed action and when they perform the same action [21-23] —suggests they possess the neural framework for perception and action that is associated with imitation. However, can monkeys also imitate by "learning to do an act from seeing it done" [24], restricted to the acquiring behaviors novel to the individual's repertoire (the 'transfer of skill' problem)? It has been suggested that in order for a response to be considered acquired through imitation it must be novel [25]. The behavior can be thought of as novel if the probability of the behavior is low at the start of the experiment and an increase in the behavior cannot be attributed to priming, motivational or attentional effects [10, 26].

2 IMITATION IN COMMON MARMOSETS

We have focused on one species of New World monkeys of the family *Callithrichidae*, the common marmoset (*Callithrix jacchus*). Callitrichid monkeys are small monkeys once thought to have retained many primitive primate characters and to be rather unsophisticated [27]. Therefore, marmosets and tamarins would not seem likely candidates for studies of complex cognition. However, this evaluation has changed [for a review, see 28], and it is currently accepted that they have developed a number of remarkably original adaptations for their unusual lifestyle [29]. More than this, Callitrichids are likely to locate food by using some sort of cognitive map [30], represent objects and their movements in an abstract manner [31], and benefit from social influences that aid in learning about new food by motivational and perceptual factors [32]. Marmosets and tamarins are remarkably sensitive and responsive to cues from other social companions, especially in the third and fourth month of life [33]. Their high level of tolerance

[1] University of Vienna, Vienna, Austria, email: ludwig.huber@univie.ac.at

during group foraging and also their sharing of food in both passive and active manners [34, 35] may be related to their cooperative breeding system [29]. All together, these social features may be responsible for their high degree of maintaining spatial and behavioral cohesion with their social partners in comparison to Capuchin monkeys (*Cebus* sp.). They are also more neophobic than capuchins, less likely to explore new places and, therefore, less likely to explore new foods or to solve new manipulative problems on their own [36].

These behavioral and social aspects of callithrichids inspired a number of experimental studies focusing on the ability of common marmosets to learn from conspecific demonstrators an food-processing technique [37-39]. All three studies used variants of the same experimental procedure (non-observer control): First, subjects (observers) were allowed to observe a physically separated conspecific (demonstrator) opening a novel apparatus – the "artificial fruit" [40] – in order to retrieve food from it, and thereafter these subjects themselves were given the opportunity to manipulate the apparatus on their own. The subjects' behavior was then compared with naïve animals that were confronted with the apparatus without prior observation of conspecifics (non-observers), and – in the Voelkl and Huber study [39] – also with observers which saw another demonstrator opening the apparatus in a different way (two-action procedure).

In the first study of this kind, Bugnyar and Huber [37] presented common marmosets a box with a pendulum door that could be either pushed or pulled to gain access to food inside the box. Observers were allowed to watch a conspecific demonstrator pull open the door. The observers showed less exploratory behavior than non-observers and, most importantly, two of them showed a strong tendency to use the demonstrated opening technique in the initial phase of the test. Only after some trials, in which they acquired own experience of opening the pendulum door, did they begin to perform the simpler solution of pushing, which was preferred by the non-observers. The authors argued that pulling the door in order to get access to food was not a simple act but a compound action-pattern. The authors distinguished four independent elements plus one dependent element in the pulling performance of the demonstrator: (1) using the left hand, (2) taking the door from the right gap, (3) pulling, (4) holding the door wide open with one hand, and (5) taking the food. Two observer marmosets copied all of these actions in the appropriate order, which is very unlikely to be due to chance, considering the combined probability for spontaneous occurrence of these actions.

In an attempt to provide data allowing a direct comparison between species, Caldwell and Whiten [38] used a marmoset-sized version of an artificial fruit that has been designed for studies of imitation in children and chimpanzees [41]. One demonstrator ('full' demonstrator) was trained to open the apparatus by removing a handle, while the other demonstrator ('partial' demonstrator) simply ate food from the lid of the apparatus. Unfortunately, none of the observers was successful in opening the apparatus – probably due to the technical sophistication of the opening mechanisms. However, the authors found clear response differences consistent with the different demonstration modes. Those animals that watched the 'partial' demonstrator performing predominantly mouthing behaviors used their mouth more frequently, while those that watched the 'full' demonstrator showing predominantly hand manipulation used their hands more frequently. The authors described these findings as body part copying, but they pointed out that the behavior of the observers might have been dependent on several other social learning effects as well. For instance, it may be the case that reaching or grasping behaviors are in some way contagious (i.e., triggered by the same response) in marmosets or that the fact that the movement of the apparatus was clearly different for both observer groups could account for the social learning seen.

Only a *two-action method*, which involves two demonstrators that differ in their body movements but create the same changes in the environment, controls for learning about the changes of state in the environment and therefore provides the most convincing evidence yet for imitative learning in animals [10, 42, 43]. Voelkl and Huber [39] applied this methodology, permitting two groups of marmosets to observe a demonstrator using one of two alternative techniques to remove the lids of baited film canisters and compared their initial test responses with one another and with a third group of marmosets that were never given the opportunity to observe a demonstrator. Furthermore, while one technique involved hand-opening common to marmosets, the other technique consisted of a behavioral 'peculiarity' (mouth-opening); that is, mouth opening was neither common in the animals under investigation nor necessary for lid removal. This requirement ensured that if the observers performed the technique, then they were most probably influenced by what they witnessed.

In fact, both groups of observers preferred to open the canisters using the same method as their demonstrator. Since hand and mouth demonstrators brought about identical changes to the canister (opening and exposing the food reward), the differential test behavior of the animals suggests that they indeed learned something about the demonstrator's behavior, rather than about certain properties of the canister. Furthermore, non-observers rarely opened the canister with the mouth, but they opened as many canisters as did members of both observer groups. An actual benefit to observer animals in terms of success rate could be found when the task was made more difficult by closing the lids of the canisters much more firmly. After this change, only the mouth-openers achieved opening the canisters and retrieving the desired mealworms. Thus, even 'slavish' copying (i.e., copying in the absence of insight) may therefore have beneficial effects for observers [26]. Furthermore, as emphasized by Caldwell and Whiten [38], social learning may provide particular practical benefits to individuals when it induces an individual to persist with unrewarded manipulations of an object, as individual learning (trial and error) is unlikely to be successful under such circumstances.

Although this study implies that learning by imitation is

more widespread in the animal kingdom than recently assumed, both the studies that failed to demonstrate imitation in monkeys as well as our own experiences in previous studies suggests that imitation is a rarely used mechanism. This might be due to the special requirements of imitative learning. To allow detailed observations imitation requires proximity of the individuals. Thus imitation is more likely to be found in species with egalitarian social systems where the individual distances between all group members are small. Additionally the copied behavior itself must show certain characteristics to be appropriate for transmission by imitation. If the behavior in question is to simple it is more likely that it is learned by individual learning, while if the behavior is on the other hand to complex it is unlikely that it can be learned by observational learning at all. A further limiting factor in imitative learning – but widely neglected in discussions and experiments until today – is the attention of the observer. If the observer does not pay attention to the whole action or action sequence demonstrated, but looses interest before it is completed, it will not be able to learn the action sufficiently well or at all. The attention-holding process might vary according to the dominance, sex and relationship of the demonstrator, as well as the type of action demonstrated, and the vigilance, interest, and motivation of the observer. While we found generally quite short attention spans in marmosets, they are sufficiently long for the actions demonstrated in our experiments [44]. However, in the study with the film canisters, the observers have been found to be particularly attentive.

3 ACTION IMITATION

Recent theories of imitation have dissected the imitative act into two components, the body part used and the action performed [45]. With respect to *body part imitation*, the finding that marmosets would copy mouth versus hand use [39] and pigeons likewise would copy beak versus food use [46] was important in classing these as true imitation [9-11, 42, 47, 48]. Use of different body parts to deal with the same task relies on visuomotor mapping from seen parts of the model's body to equivalent parts of the self. But for an action to qualify as imitation in the restricted sense of "learning how to move" (*action imitation*), the observer must learn the specific response topography, i.e., the specific action by which the response is made [9]. Despite various approaches to, and definitions of, imitation [2-4], most scholars agree that when an individual replicates an action that it has observed being performed by another individual it requires a matching system that allows conversion of observed actions by others into actions executed by oneself. In other words, visual input needs to be transformed into corresponding motor output (the 'correspondence' problem).

The greatest challenge for an animal solving the correspondence problem is to perform imitation of 'perceptually opaque' actions, those model actions of which the observer's image is not similar to the sensory feedback received during performance of the same action [42]. This is particularly true if the action demonstrated by the model does not already exist in the observer's behavioral repertoire. So far, precise copying of novel actions is underspecified in theory and vague in evidence. The models currently available, including those that rest on mirror neurons, are not sufficiently competent to explain high matching fidelity in the imitation of *novel* actions, thereby solving both problems of imitation, the transfer of skill and the correspondence problem, at the same time. There is also no convincing evidence of movement copying in nonhuman animals with the trajectory of the movement or the dynamics of the model's action being replicated by the observer with high fidelity. The few cases in which animals have been reported as achieving some degree of matching are lacking rigorous quantitative analysis of the matching degree (e.g. only qualitative descriptions of the imitator's performance are provided [49, 50].

A paradigm that has come closest to the assessment of the precision of copying is the "do-as-I-do paradigm". I a replication of the classic study by Hayes and Hayes [51], Custance and colleagues [52] found only a modest degree of matching between tutor and subject. Coders blind to what each chimpanzee has actually watched identified some matching in relation to touching several parts of the body in sight, as well as out of sight, symmetric and asymmetric conjunctions of hands, digit movements, and hand or whole-body actions. However, the matching fidelity was low overall; only a small fraction of the novel actions were reproduced (13 or 20 from a total of 48 novel gestures), and the imitations were far from perfect. Similarly, in a study by Myowa-Yamakoshi [53], five female nursery-reared chimpanzees rarely reproduced a demonstrated action at the first attempt (less than 6% of the overall actions). In a further, more recent study, only 20% of the chimpanzee observers matched the demonstrator's actions, i.e., opening a tube with the hands [54].

4 THE PRECISE COPYING OF MOVEMENT TRAJECTORIES

To investigate imitation of movement patterns in marmoset monkeys we reanalyzed the actions shown by the mouth model and her six observers of the Voelkl and Huber [39] study and – for the sake of comparison – tested further 24 naïve animals (non-observers) that had never observed a model. In order to video-capture the movements from a fixed perspective, only one baited canister was attached to a wooden board that was placed in a six cm gap between a glass window in the front-side of the testing cage and an opaque partition wall. This setup ensured that animals could approach the artificial fruit from only two directions – in both cases approximately parallel to the window and the lens of the video camera. The completely shut canister required – due to the tightness of the lid – a powerful opening technique. Our marmosets have never achieved to open a completely shut canister by hand but only by mouth.

The model, five out of six observers, but only four out of 24 non-observers succeeded in opening the containers with their mouth (Voelkl & Huber, in prep.). These opening attempts provided six opening movements of the model, 14 of observers and 21 of non-observers. The head movement

of the subjects was tracked by manually identifying the position of five morphological features in the face of the subject on a frame-to-frame basis (25 frames per s). We then defined five parameters to describe the movement, used discriminant function analysis of the orthogonalized data, and thus generated a function with clearly distinctive mean discriminant scores for movements of the model and the non-observers. As main result, we found the mean scores for the observers being closer to the mean of the model than to the non-observer. Thirteen out of 14 observer movements were classified as model movements. Thus, the movement patterns of the observers were clearly more similar to the movement pattern of the model than to those of the non-observers.

5 HOW DO BRAINS SOLVE THE IMITATIVE LEARNING PROBLEM?

More than a century of research on imitation has left us with a crucial functional problem: how are observers able to transform a visual presentation of an action into motor output? For most currently available theories of imitation the key to solution is automatic activation of *existing* motor representations. But here marmosets learned by observation a novel movement pattern, not available from the own behavior repertoire. Even if we assume that marmosets have already performed similar movements before, like biting into an object or levering it up with the head, how can we explain the exact matching of the observers, e.g. the convergence of the paths of the head, the same inclination of the head in the course of the opening action? A minimal requirement would be to adjust an action present in one's motor repertoire to a different observed action [55].

As evidenced by the significant difference in the movement shown by observers and non-observers, opening a film box is not an all-or-nothing behavior for marmosets. There are still many degrees of freedom for the exact performance, created by the movements of the head and the whole body when attempting to open the lid of the film canister. The common problem for imitation theories is to account for the close convergence of Bianca's (the model) and the observers' opening actions despite the actual variance of ways to achieve the common goal of opening the lid as evidenced by the non-observers. Which of the many theories currently available in the literature can offer a sufficient explanation of the creation of a novel action from using only visual information? Or more generally, what does this result tell us about what is actually involved in successful action imitation, i.e., learning how to move the body by observing the behavior of others? And what role do the mirror neurons play?

Mirror neurons might code the likely future actions of others so that observers are able to anticipate the intention of others [56] rather than to provide a form of motor learning. Macaques, for instance, might have used their mirror neurons to recognize being imitated [19]. However, the actions that they were shown by the human model have already existed in the observers' motor repertoire. Perhaps a first step in the direction of clarifying the potential role of mirror neurons for imitative learning might be the detection of a new type of visuomotor neurons, called *tool-responding mirror neurons*, in the lateral section of the macaque monkey's ventral premotor area F5 [57]. The neurons show experience-dependent responses and perhaps enable the observing monkey to extend action-understanding capacity to actions that do not strictly correspond to its motor representations. However, in contrast to our findings, these neurons were found to discharge only after a relatively long visual exposure to actions of a tool-using experimenter. It was therefore proposed that the changes in the body schema and/or in the motor representations of the observer are possible only for motor training [58], but that tool actions cannot be directly translated into own motor repertoire. The authors concluded with hypothesizing that a mirror mechanism evolved in monkeys for action understanding, but only emerged in human evolution as suitable neural substrate for imitation [57].

Recently, theoretical and empirical attempts have been made to explain imitative learning through reafferent feedback loops in the brain. As part of a conceptual framework for motor learning and sensorimotor control, the 'modular selection and identification for control model' (MOSAIC) is based on multiple pairs of 'predictor' and 'controller' models processing feedforward and feedback sensorimotor information, respectively [59-61]. Indeed, the MOSAIC model has been shown to learn a simple acrobat task (swinging up a jointed stick to the vertical) through action observation and imitation [62]. The results of functional magnetic resonance experiments suggested the superior temporal sulcus (STS) as the region at which the observed actions, and the reafferent motor-related copies of actions made by the imitator, interact [63]. Furthermore, in the macaque there seems to be a circuitry composed of the STS, providing a higher-order visual description of the observed action, the rostral sector of the inferior parietal lobule (PF) and the ventral premotor cortex (area F5) that codes the action of others and maps it onto the motor repertoire of the observer [64]. Thus, imitative learning is supported by interaction of the core circuitry of imitation with the dorsolateral prefrontal cortex and perhaps motor preparation areas — namely, the mesial frontal, dorsal premotor and superior parietal areas. In humans, this direct route of visuo-motor conversion on a sensory-motor level of imitation is used especially for transforming a novel or meaningless action into a motor output, while a semantic mechanism, working on the basis of stored memories, allows reproductions of known actions on an intentional level of processing [65-67]. It remains to be shown whether non-human animals can also use multiple routes of action imitation.

In conclusion, the present findings suggest that monkeys are not only able to reproduce known actions shown by others or to recognize when others reproduce actions they themselves have executed before, but are also able to learn new actions by observation. Such abilities are functional by providing a type of learning that avoids remaining with insufficient or ineffective variants or time-consuming trial-and-error learning.

ACKNOWLEDGEMENTS

This work has received funding from the Austrian Science Fund (P12011-BIO) and the European Community (NEST 012929).

REFERENCES

1. Byrne, R.W., *Imitation as behaviour parsing.* Philos Trans R Soc Lond B Biol Sci, 2003. **358**(1431): p. 529-36.
2. Heyes, C.M. and B.G.J. Galef, eds. *Social Learning in Animals: The Roots of Culture*. 1996, Academic Press: San Diego.
3. Meltzoff, A.N. and W. Prinz, eds. *The imitative mind: development, evolution, and brain bases*. 2002, Cambridge University Press: Cambridge, UK. 353.
4. Hurley, S. and N. Chater, eds. *Perspectives on Imitation: From Neuroscience to Social Science - Volume 1: Mechanisms of Imitation and Imitation in Animals*. 2005, MIT Press: Cambridge, MA.
5. Zentall, T.R. and B.G. Galef, Jr., eds. *Social learning: Psychological and biological perspectives*. 1988, Erlbaum: Hillsdale, New Jersey.
6. Byrne, R.W., *The evolution of intelligence*, in *Behaviour and Evolution*, P.J.B. Slater and T.R. Halliday, Editors. 1994, Cambridge University Press: Cambridge. p. 223-264.
7. Byrne, R.W. and A.E. Russon, *Learning by imitation: a hierarchical approach.* Behavioral and Brain Sciences, 1998. **21**(5): p. 667-84; discussion 684-721.
8. Heyes, C.M., *Social learning in animals: categories and mechanisms.* Biological Reviews, 1994. **69**: p. 207-231.
9. Zentall, T., *Imitation by animals: how do they do it?* Current Directions in Psychological Science, 2003. **12**(3): p. 91-95.
10. Zentall, T.R., *Imitation in animals: evidence, function, and mechanisms.* Cybernetics and Systems: An International Journal, 2001. **32**: p. 53-96.
11. Whiten, A., et al., *How do apes ape?* Learning and Behavior, 2004. **32**(1): p. 36-52.
12. Tomasello, M. and J. Call, *Primate cognition*. 1997, Oxford: Oxford University Press.
13. Visalberghi, E. and D.M. Fragaszy, *Do monkeys ape?*, in *"Language" and intelligence in monkeys and apes. Comparative developmental perspectives*, S.T. Parker and K.R. Gibson, Editors. 1990, Cambridge University Press: Cambridge. p. 247-273.
14. Visalberghi, E. and D.M. Fragaszy, *''Do monkeys ape?'' Ten years after*, in *Imitation in animals and artefacts*, K. Dautenhahn and C. Nehaniv, Editors. 2002, MIT Press: Cambridge. p. 471–499.
15. Kawai, M., *Newly-acquired pre-cultural behaviour of the natural troop of Japanese monkeys in Koshima Islet.* Primates, 1965. **6**: p. 1-30.
16. Hirata, S., K. Watanabe, and M. Kawai, *"Sweet-potato washing" revisited*, in *Primate origins of human cognition and behavior*, T. Matsuzawa, Editor. 2001, Springer: Tokyo. p. 487-508.
17. Galef, B.G.J., *The question of animal culture.* Human Nature, 1992. **3**: p. 157-178.
18. Subiaul, F., et al., *Cognitive imitation in rhesus macaques.* Science, 2004. **305**(5682): p. 407-410.
19. Paukner, A., et al., *Macaques (Macaca nemestrina) recognize when they are being imitated.* Biology Letters, 2005. **1**: p. 219-222.
20. Ferrari, P.F., et al., *Neonatal imitation in rhesus macaques.* PLoS Biol, 2006. **4**(9): p. e302.
21. Gallese, V., et al., *Action recognition in the premotor cortex.* Brain, 1996. **119 (Pt 2)**: p. 593-609.
22. Rizzolatti, G., et al., *Premotor cortex and the recognition of motor actions.* Brain Res Cogn Brain Res, 1996. **3**(2): p. 131-41.
23. Rizzolatti, G. and L. Craighero, *The mirror-neuron system.* Annu Rev Neurosci, 2004. **27**: p. 169-92.
24. Thorndike, E.L., *Animal intelligence: An experimental study of the associative processes in animals.* Psychol. Rev. Monogr. Suppl., 1898. **2**: p. 79-.
25. Thorpe, W.H., *Learning and instinct in animals.* 1956, London: Methuen.
26. Huber, L., *Movement imitation as faithful copying in the absence of insight.* Behavioral and Brain Sciences, 1998. **22** (5): p. 694.
27. Hershkovitz, P., *Living New World Monkeys (Platyrrhini), with an Introduction to Primates. Vol. 1*. 1977, Chicago: Chicago University Press.
28. Huber, L. and B. Voelkl, *Social and physical cognition in marmosets and tamarins*, in *The Smallest Anthropoids: The Callimico/Marmoset Radiation*, S.M. Ford, L.C. Davis, and L. Porter, Editors. in press, Springer: New York.
29. Snowdon, C.T., *Social processes in communication and cognition in callitrichid monkeys: a review.* Animal Cognition, 2001. **4**: p. 247-257.
30. Garber, P. and P. Hannon, *Modeling monkeys: a comparison of computer-generated and naturally occurring foraging patterns in two species of Neotropical primates.* International Journal of Primatology, 1993. **14**: p. 827-852.
31. Mendes, N. and L. Huber, *Object Permanence in Common Marmosets (Callithrix jacchus).* Journal of Comparative Psychology, 2004. **118**(1): p. 103-112.
32. Voelkl, B., C. Schrauf, and L. Huber, *Social contact influences the response of infant marmosets towards novel food.* Animal Behaviour, 2006. **72**(2): p. 365-372.
33. Schiel, N. and L. Huber, *Social influences on the development of foraging behavior in free-living common marmosets (Callithrix jacchus).* American Journal of Primatology, 2006. **68**: p. 1-11.
34. Feistner, A.T.C. and E.C. Price, *Food offering in new world primates: two species added.* Folia Primatologica, 1991. **57**: p. 165-168.
35. Ferrari, S.F., *Food transfer in a wild marmoset group.* Folia Primatologica, 1987. **48**: p. 203-206.
36. Fragaszy, D.M. and E. Visalberghi, *Socially biased learning in monkeys.* Learning and Behavior, 2004. **32**: p. 24-35.
37. Bugnyar, T. and L. Huber, *Push or pull: an experimental study on imitation in marmosets.* Animal Behaviour, 1997. **54**(4): p. 817-831.
38. Caldwell, C.A. and A. Whiten, *Testing for social learning and imitation in common marmosets, Callithrix jacchus, using an artificial fruit.* Anim Cogn, 2004. **7**(2): p. 77-85.
39. Voelkl, B. and L. Huber, *True imitation in marmosets.* Animal Behaviour, 2000. **60**(2): p. 195-202.
40. Whiten, A., et al., *Imitative learning of artificial fruit processing in children (Homo sapiens) and chimpanzees (Pan troglodytes).* Journal of Comparative Psychology, 1996. **110**: p. 3-14.
41. Whiten, A., et al., *Imitative learning of artificial fruit processing in children (Homo sapiens) and chimpanzees (Pan troglodytes).* J Comp Psychol, 1996. **110**(1): p. 3-14.
42. Heyes, C.M. and E.D. Ray, *What is the significance of imitation in animals?*, in *Advances in the Study of Behavior*, P.J.B. Slater, et al., Editors. 2000, Academic Press: New York. p. 215-245.

43. Whiten, A. and R. Ham, *On the nature and evolution of imitation in the animal kingdom: Reappraisal of a century of research*, in *Advances in the study of behavior*, P.J.B. Slater, et al., Editors. 1992, Academic Press: New York. p. 239-283.
44. Range, F. and L. Huber, *Attention span of common marmosets - Implications for social learning experiments.* Animal Behaviour, in press.
45. Chaminade, T., A.N. Meltzoff, and J. Decety, *An fMRI study of imitation: action representation and body schema.* Neuropsychologia, 2005. **43**(1): p. 115-27.
46. Zentall, T.R., J.E. Sutton, and L.M. Sherburne, *True imitative learning in pigeons.* Psychological Science, 1996. **7**(6): p. 343-346.
47. Miklósi, A., *The ethological analysis of imitation.* Biological Reviews, 1999. **74**: p. 347-374.
48. Heyes, C.M., *Causes and consequences of imitation.* Trends in Cognitive Science, 2001. **5**: p. 253–261.
49. Moore, B.R., *Avian movement imitation and a new form of mimicry: tracing the evolution of a complex form of learning.* Behaviour, 1992. **122**(3-4): p. 231-263.
50. Tayler, C.K. and G.S. Saayman, *Imitative behaviour by Indian Ocean bottlenose4 dolphins (Torsiops aduncus) in captivity.* Behaviour, 1973. **44**: p. 286-298.
51. Hayes, K.J. and C. Hayes, *Imitation in a home-raised chimpanzee.* Journal of Comparative and Physiological Psychology, 1952. **45**: p. 450-459.
52. Custance, D.-M., A. Whiten, and K.A. Bard, *Can young chimpanzees (Pan troglodytes) imitate arbitrary actions? Hayes & Hayes (1952) revisited.* Behaviour, 1995. **132**(11-12): p. 837-859.
53. Myowa-Yamakoshi, M., *Evolutionary foundation and development of imitation*, in *Primate origins of human cognition and behavior*, T. Matsuzawa, Editor. 2001, Springer: Tokyo.
54. Call, J., M. Carpenter, and M. Tomasello, *Copying results and copying actions in the process of social learning: chimpanzees (Pan troglodytes) and human children (Homo sapiens).* Anim Cogn, 2005. **8**(3): p. 151-63.
55. Rizzolatti, G., *The mirror neuron system and imitation*, in *Perspectives on imitation. From neuroscience to social science*, S. Hurley and N. Chater, Editors. 2005, MIT Press: Cambridge, MA. p. 55-76.
56. Fogassi, L., et al., *Parietal lobe: from action organization to intention understanding.* Science, 2005. **308**(5722): p. 662-7.
57. Ferrari, P.F., S. Rozzi, and L. Fogassi, *Mirror neurons responding to observation of actions made with tools in monkey ventral premotor cortex.* Journal of Cognitive Neuroscience, 2005. **17**: p. 212-226.
58. Iriki, A., M. Tanaka, and Y. Iwamura, *Coding of modified body schema during tool use by macaque postcentral neurones.* Neuroreport, 1996. **7**(14): p. 2325-30.
59. Haruno, M., D.M. Wolpert, and M. Kawato, *Mosaic model for sensorimotor learning and control.* Neural Computation, 2001. **13**: p. 2201–2220.
60. Wolpert, D.M., K. Doya, and M. Kawato, *A unifying computational framework for motor control and social interaction.* Philos Trans R Soc Lond B Biol Sci, 2003. **358**: p. 593-602.
61. Wolpert, D.M. and M. Kawato, *Multiple paired forward and inverse models for motor control.* Neural Networks, 1998. **11**: p. 1317–1329.
62. Doya, K., et al., *Recognition and imitation of movement patterns by a multiple predictor–controller architecture.* Technical Rep. IEICE, 2000. **TL2000-11**: p. 33–40.
63. Iacoboni, M., et al., *Reafferent copies of imitated actions in the right superior temporal cortex.* Proc Natl Acad Sci U S A, 2001. **98**(24): p. 13995-9.
64. Iacoboni, M., *Neural mechanisms of imitation.* Curr Opin Neurobiol, 2005. **15**(6): p. 632-7.
65. Rumiati, R.I. and H. Bekkering, *To imitate or not to imitate? How the brain can do it, that is the question!* Brain and Cognition, 2003. **53**: p. 479-482.
66. Rumiati, R.I. and A. Tessari, *Imitation of novel and wellknown actions: The role of short-term memory.* Experimental Brain Research, 2002. **142**: p. 425–433.
67. Rumiati, R.I., et al., *Common and differential neural mechanisms supporting imitation of meaningful and meaningless actions.* J Cogn Neurosci, 2005. **17**(9): p. 1420-31.

Visuo-Cognitive Perspective Taking for Action Recognition

Matthew Johnson and **Yiannis Demiris** [1]

Abstract. Many excellent architectures exist that allow imitation of actions involving observable goals. In this paper, we develop a Simulation Theory-based architecture that uses continuous visual perspective taking to maintain a persistent model of the demonstrator's knowledge of object locations in dynamic environments; this allows an observer robot to attribute potential actions in the presence of goal occlusions, and predict the unfolding of actions through prediction of visual feedback to the demonstrator. The architecture is tested in robotic experiments, and results show that the approach also allows an observer robot to solve Theory-of-Mind tasks from the 'False Belief' paradigm.

1 Introduction

When we see another person performing an action, we are usually able to understand the purpose and intention underlying the action, and can reproduce the action for ourselves. The HAMMER architecture [5, 16] can be used to equip a robot with this common human ability. The HAMMER architecture achieves the mapping between observed and self-generated action by directly involving the observer robot's motor system in the action recognition process; during observation of the demonstrator's actions, all the observer's inverse models (akin to motor programs) are executed in parallel in simulation using forward models. The simulated actions generated by the inverse models are compared to the observed action, and the one that matches best is selected as being the observed action. The internal action simulation, combined with the comparison to the observed action, achieves the mapping between observed action and self-generated action that is required for imitation [4].

By using the motor system to achieve action recognition, the HAMMER architecture is taking a Simulation Theory approach to solving the imitation problem. In the 'Theory of Mind' paradigm, the Simulation Theory is used to attribute mental states to other people by using one's own cognitive decision-making mechanism as a manipulable model of other's minds, taken off-line and placed into the context of their situation [13, 9, 8]. For this to work, the state of the 'target' agent is used instead of one's own state, but transformed into an egocentric format that our first-person decision-making and behaviour-generation mechanisms will accept.

Similarly, in order to provide meaningful data for comparison, the simulated actions used by the HAMMER architecture during recognition must be generated as though from the point of view of the demonstrator. Since the HAMMER architecture uses a Simulation Theory approach, the observer's inverse models require first-person data in order to generate actions, and so spatial and visual perspective taking are used to achieve the egocentric 'shift' from the observer to the demonstrator. The data required for the inverse models to operate is therefore derived from consideration of the demonstrator's physiospatial and sensory circumstances, and not the observer's, using perspective taking [11].

However, it is not only instantaneous sensory information that informs goal selection and action planning. It is through keeping in memory details of objects that are seen that *cognitive maps* are built up, which are critical to action generation. In this paper, we present a Simulation Theory approach to perspective taking that allows an observer robot to use its visual perceptual mechanisms in simulation to determine what the demonstrator is seeing; by performing this process continually, the observer's first-person cognitive map generation routines can be used to build up and maintain a representation of the demonstrator's own cognitive map. Taking into account the demonstrator's knowledge of the world in this manner allows more accurate state and goal information to be fed to the HAMMER architecture.

2 Background

In common and also academic use, the term 'perspective taking' has many meanings in many different situations. There are such definitions as:

- "People's ability to experience and describe the presentation of an object or display from different vantage points" [1]
- "Imagining oneself in another's shoes" [7]
- "Understand[ing] how others perceive space and the relative positions of objects around them- [...] the ability to see things from another person's point of view" [15]
- "Consider[ing] the needs and wants of the opponent" [6]

In this paper we focus on equipping robots with perceptual and cognitive perspective-taking abilities, through a Simulation Theory approach, in order to improve the quality of the state information fed to the HAMMER architecture. In this architecture, a cognitive map is defined as being a *representation in memory of the location of observed objects*. This memory is updated continually from observation of the environment, and is available to the action generation system for action planning. The cognitive map is used also as a manipulable spatial model of the environment to facilitate perspective taking; to enable visual perceptual perspective taking, the objects in the cognitive map are linked with visuo-spatial representations that are used to re-create the visual image seen by the demonstrator.

[1] BioART, ISN Group, Department of Electrical and Electronic Engineering, Imperial College London. Email: {matthew.johnson, y.demiris}@imperial.ac.uk

2.1 HAMMER

The HAMMER (Hierarchical Attentive Multiple Models for Execution and Recognition) architecture is a Simulation-Theoretical architure for action recognition and imitation, based on the hierarchical coupling of internal models to produce simulation loops. HAMMER achieves first-person action generation using coupled *inverse* and *forward* models, and uses the same arrangement to achieve imitation, but fed from a perceptual perspective-taking process involving internal inverse and forward *vision models*. The perceptual perspective taking process has been shown to improve the performance of action recognition in situations where the observer must take into account visual occlusions and visual cues provided to the target [11].

2.2 Internal Inverse and Forward Models

One of the core components of the HAMMER architecture is the *inverse model* for motor control. Inverse models represent functionally specialised units for generating actions to achieve certain goals. The generic inverse model takes as input the *current state* of a system, a *goal state* that is the system's desired state, and produces as output the action required to move the system from its current state to the goal state [12, 18]. In the control theory literature, the inverse model is known as a *controller* and its outputs are control signals; when applied to robotics, the current state is the state of the robot and its environment, and the outputs are motor commands. In that context, inverse models are known as *behaviours*.

Inverse models have several *internal states*, that are used in action execution and recognition [10]. One of these states is the *applicability* of the inverse model. When presented with a goal, the inverse model will calculate its level of applicability through simulation with its coupled forward model. The applicability is a measure of how useful the inverse model is for achieving the goal. A low applicability level means that the inverse model cannot achieve the goal from its current state, for example, the "Place object on shelf" inverse model when the shelf is too high to reach. The applicability level is explained in more detail in Section 3.3.

Forward models of causal dynamics are used in predictive control systems. The classic forward model takes as input a system state and the dynamics currently acting on the system, and produces as output the *predicted next state of the system*. In the HAMMER architecture, multiple forward models are *coupled* to inverse models to create a simulation process. This approach is similar to that used in other internal model-based systems for motor control [21, 20]. When coupled to an inverse model, a forward model receives the action output from the inverse model; the forward model then generates a prediction of the state that would result, if the action was to be performed.

2.3 Inverse and Forward Vision Models

In [11], the capacity for *visual perceptual perspective taking* was introduced to the HAMMER architecture. In keeping with the simulation theoretical approach, this was achieved through a biologically inspired *simulation of visual perception*. In the same way as action recognition and imitation is achieved in the HAMMER architecture through coupled inverse and forward models as used in control, visual perception and perspective taking is performed here through coupled inverse and forward models of the *visual* process. The *inverse vision model* is defined as taking two inputs, the first being a camera image, and the second being the visual parameters with which to process that image. The output from the model is the state

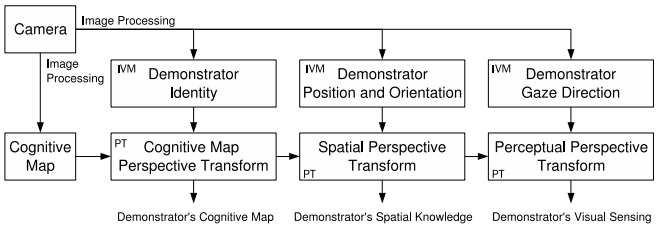


Figure 1. The perspective-taking process. Image information from the camera, and the robot's own knowledge held in the cognitive map, are fed into a cascade of perspective transform 'filters'. The outputs at each stage are used as the 'pretend states' fed into the HAMMER architecture. 'PT' indicates a perspective transform stage, and 'IVM' indicates an inverse vision model for performing image processing.

output from processing. A *forward vision model* is defined as having two inputs and one output. The forward vision model takes as input visual object properties retrieved from the cognitive map (e.g. colour, shape, etc), and their desired state (e.g. positions and orientations taken from the cognitive map), and produces as output the visual image that results from reconstructing these inputs. Inverse and forward vision models are described in detail in [11].

The coupling of inverse and forward vision models results in simulation of perception, and gives HAMMER the ability to consider what the demonstrator *sees*, as well as its position. This enables the observer robot to take into account visual occlusions effecting the demonstrator, and through *continual* usage, the observer can keep track of what objects the demonstrator has seen in the past and potentially stored in its cognitive map. Because the demonstrator sees different things to the observer due to its differing viewpoint, it becomes necessary for the observer robot to maintain a representation of the demonstrator's cognitive map in order to predict and recognise actions. In keeping with the simulation theory approach, this may be achieved by recruiting the observer's own cognitive map creation and updating processes, but fed with information derived from visual perceptual perspective taking instead of first-person visual information. Figure 2 shows the perception simulation process.

3 Implementation

The perspective taking architecture shown in Figures 1 and 2 was implemented in C++ for experiments involving an observing observer robot and a demonstrator robot. The target robots were ActivMedia Peoplebots, equipped with grippers and firewire cameras. A version of HAMMER was implemented and linked to the perspective taking architecture.

3.1 Inverse and Forward Vision Models

Inverse vision models were implemented using the ARToolkit Plus, an extension of the ARToolkit [2]. When presented with an image containing two-dimensional square markers (fiducials) of known size, the ARToolkit can calculate the position and orientation of the markers in world co-ordinates. A set of three objects was therefore produced with fiducials attached, and in order to extract the demonstrator robot's position and orientation at any point in time, a cubic AR 'hat' was made, with a fiducial on each vertical face. This ensured that no matter which direction the demonstrator robot was fac-

ing, the observer robot would be able to determine its location and orientation.

To construct visual scenes from the transformed cognitive map, the forward vision models used the OpenGL graphics library (www.opengl.org). To ensure that the same inverse vision models as used for first-person visual processing would work with the reconstructed image for the demonstrator robot, the fiducials used by the ARToolkit were added in as OpenGL textures and linked to the object entries in the cognitive map.

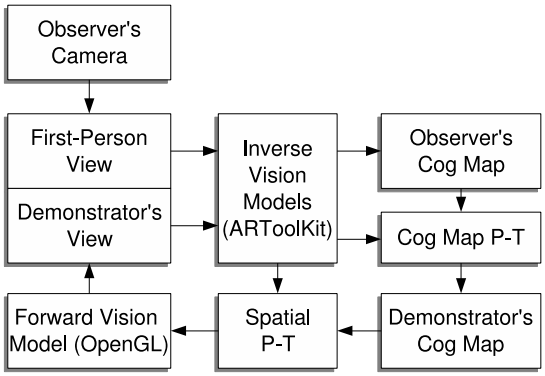


Figure 2. The perception simulation loop. The observer's first-person view of the scene is used to build up the observer's cognitive map of the scene. The cognitive map is filtered through the cognitive map perspective transform that 'filters' the observer's cognitive map to make it like the demonstrator's. This is then used as a basis for the perceptual perspective transform, that begins with a spatial geometric transform to 're-centre' on the demonstrator, and then fed through the forward vision model to re-create what the demonstrator is seeing. The observer can then use its inverse vision models on the image to update its representation of the demonstrator's cognitive map, in the same way as it would update its own.

3.2 Cognitive Map Definition

The cognitive map was defined as being a list of objects, held in memory. It was assumed that the robots already knew what each object was and could identify them through the inverse vision models (i.e. the inverse vision models were programmed to recognise the objects, through use of the fiducials, and extract relevant information). When visual information for the objects was available from the inverse vision models, the cognitive map entries for those objects were updated with world position and orientation values. Linked with this information was a three-dimensional model of each object, and visual information (e.g. colour and texture) that would be used by the observer's forward vision model to re-create the image of the scene from the point of view of the demonstrator. As can be seen from Figure 1, the perspective-taking process is then comprised of the following steps:

1. The demonstrator is identified and the correct cognitive map perspective transform, comprising the differences from the observer's own cognitive map, is applied;
2. A spatial perspective transform is applied to the resulting cognitive map, to re-centre it upon the demonstrator;
3. The forward vision model takes in the re-centred spatial data, and accesses the visual information linked to the objects in the cognitive map, to re-construct the image that the demonstrator is seeing. This image is then processed by the observer's inverse vision models, to update the demonstrator's cognitive map transform, and to provide state information to HAMMER.

3.3 Inverse and Forward Models

Inverse models for the HAMMER architecture were implemented as PID controllers, generating robot velocities and delta-angle headings in order to minimise the distance between the robot grippers and a goal object. The state information required for the inverse models was taken from either the observer's own cognitive map, or its representation of the demonstrator's cognitive map. When used for action simulation, the applicability level A_t of the inverse model was calculated for the n^{th} simulation iteration according to:

$$A_{t,n} = \begin{cases} 0 & \text{at } n = 0 \\ A_{t,n-1} + \frac{1}{S_d} \times \frac{1}{n} & \text{if } \frac{dS_d}{dt} < 0 \\ A_{t,n-1} - \frac{1}{S_d} \times \frac{1}{n} & \text{if } \frac{dS_d}{dt} \geq 0 \end{cases} \quad (1)$$

The applicability accumulation is discounted over time and is increased (rewarded) if the inverse model is making progress towards achieving its goal, and decreased (punished) if it is not. The state distance between current state S_t and the goal state vector λ is defined as:

$$S_d = \sum_{i=1}^{M} |\lambda_i - S_{t,i}| \quad (2)$$

When S_d was less than a completion threshold ϵ_1, the inverse model became complete and did not generate motor commands even when instructed to execute. In the following experiments, ϵ_1 was chosen to be 0.04.

Forward models used Euler integration to form 1-timestep predictions based on the current state, and the robot velocity and heading motor commands generated by the inverse models. As in [11], the forward models were equipped with collision models of the robot and objects in the environment, allowing the forward model to predict position and velocity states in situations when the robot ran into tables or other objects.

3.4 Perspective Taking Visual Representations — 'Ghosts'

Through coupling the perception simulation loop to the simulation of action enabled by the HAMMER architecture, it is possible to *predict the visual feedback* arising from the action. By processing this visual feedback with the inverse vision models, and updating the cognitive map representation, the action simulation can continue further into the future, and the *outcome* of actions predicted. In section 4, experiments are described in which this approach is used to create multiple *Perspective Taking Visual Representations* (PTVRs), parallel instances of the observer's own perception and action mechanisms, each driven by a different inverse model. Using perspective taking, the observer can place these 'ghosts' in the place of the demonstrator, and use them to predict what the visual feedback will be from possible actions the demonstrator may perform. This allows the observer to predict the changes to the demonstrator's knowledge of the world during the course of a possible action, and how this may effect the course of the action. Figure 3 shows the arrangement.

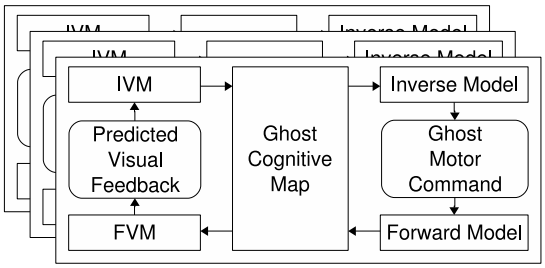


Figure 3. Perception and action simulation coupled for the generation of PTVRs ('ghosts'). Multiple ghosts can be instantiated and used in parallel, each one driven by a different inverse model. By coupling the HAMMER action simulation loop to the perception simulation loop, it is possible to predict the visual feedback to the ghost and therefore the updates to the ghost's cognitive map. A ghost can represent either the observer or the demonstrator performing a certain action.

4 Experiments

The implemented perspective-taking architecture was deployed onto the robots arranged in the scenario shown in Figure 4.

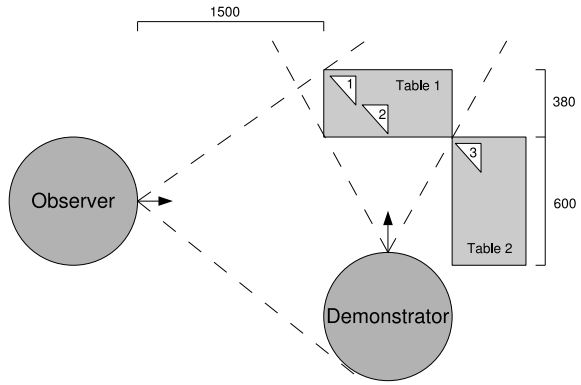


Figure 4. Plan view of the experimental setup. The observer robot faces two tables. Table 1 has objects 1 and 2, and table 2 has object 3. The demonstrator robot is positioned so that it, and the tables and the objects, can be clearly seen by the observer. The demonstrator is placed facing and close to table 1, so that initially it is able only to see objects 1 and 2. The dashed lines indicate the fields-of-view of the robots. The plan is not to scale, but measurements have been provided to indicate size and relative position. Measurements are in millimetres. The objects were 150mm across their long edge. The ARToolKit fiducials had 120mm edge, and were mounted as per Figure 5.

Three ARToolKit fiducials were attached to triangular objects, to enable both the observer and the demonstrator to identify and locate those objects. The objects were placed so that both the observer and the demonstrator could see the objects, however, initially, the observer could see all three objects whereas the demonstrator could see only the first two. The objects and the demonstrator robot were given simple 3D models to enable their reconstruction into the image produced by the forward vision model. Similarly, the implemented HAMMER architecture was provided with three 'nudge object' inverse models, one for each of the objects used during the experiments. These inverse models, when activated, produced motor commands for moving the robots from their current position to the specific object, and stopped when the robot gripper touched the object.

During the experiments it was assumed that the demonstrator's camera was kept stationary with respect to the robot's frame, pointing directly ahead of the robot.

4.1 Experimental Scenarios

The first experiment was designed to test the architecture's ability to update its cognitive map representations, in the presence of demonstrator movements, and object movements both seen and unseen by the demonstrator. There were three parts:

1. The observer takes the perspective of the demonstrator robot, and initialises its representation of the demonstrator's cognitive map;
2. Demonstrator rotates 45 degrees to its right. The observer, through continual perspective taking, updates its cognitive map representations;
3. Object 1 is moved, unseen to the demonstrator, but seen by the observer.

Experiment Two took this further, by having the observer maintain its cognitive map representations over five episodes of object and demonstrator movements, in which objects were occluded from both the demonstrator and the observer. The observer also had to use its cognitive map representations to attempt to determine what actions the demonstrator *believed* it could perform, through using perspective taking and action simulation to calculate inverse model applicabilities. The sequence was the following:

1. Demonstrator can see objects 1 and 2. Object 1 is not graspable, and the demonstrator cannot see object 3 (Figure 4);
2. Object 1 is moved close to the demonstrator, occluding object 2 from the observer;
3. Demonstrator rotates 45 degrees to its right. Object 3 becomes observable (as per Figure 7 B);
4. Object 1 is moved back to original position. The observer can see this, but the demonstrator cannot;
5. Demonstrator rotates back to original position.

In Experiment Three the observer robot had to maintain its cognitive map representations over four episodes of *simultaneous* demonstrator and object movements. The observer also had to predict the visual feedback during each potential demonstrator action using its PTVRs, in order to predict the impact of false beliefs on the performance of the actions. The sequence was the following:

1. Demonstrator can see objects 1 and 2. Object 1 is not graspable, and the demonstrator cannot see object 3 (Figure 4);
2. Demonstrator rotates 45 degrees to its right, then object 1 is moved to a graspable position (unseen by demonstrator);
3. Objects 1 and 2 are moved away (unseen by demonstrator);
4. Demonstrator rotates back to original position. Object 3 moved away (unseen by demonstrator).

5 Results

5.1 Experiment 1

Figure 6 shows the observer's view of the scene during part 1 of the experiment. The demonstrator robot and the objects are visible. Figure 6 A shows the thresholded camera image fed to the observer's inverse vision models, and Figure 6 B shows the resulting reconstruction of the visual scene, using data from the observer's cognitive

map and its forward vision model. The ARToolKit has successfully extracted the position and orientation of the objects and the demonstrator, and Table 1 shows the contents of the observer's cognitive map resulting from the processing. The X, Y, Z position and angle of objects are extracted and updated in the observer's cognitive map representations while the objects are visible.

Table 1. Cognitive map entries for centroid positions and orientations of objects when viewed by the observer robot (first-person perspective). The values shown are relative to the observer's camera position and orientation. The results correspond to the scene shown in Figure 6.

Object	**X** (m)	**Y** (m)	**Z** (m)	**Angle** (Degrees)
Demonstrator	-0.56	0.20	1.80	357.51
Object 1	0.32	-0.28	1.61	44.03
Object 2	0.20	-0.27	1.75	50.39
Object 3	-0.15	-0.24	2.21	12.13
Table 1	0.37	0.00	1.75	90.00
Table 2	-0.15	0.00	2.30	0.00

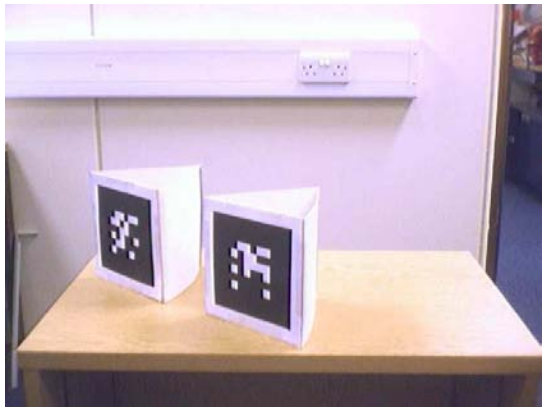


Figure 5. The demonstrator's view of the table and objects 1 and 2. Figure 7 (A) shows an internal simulation of this viewpoint by the observer.

Figure 7 shows the result of the perceptual perspective taking. Figure 7 A is what the observer determines the demonstrator to be seeing during the first part of the experiment; Figure 5 shows the demonstrator's actual camera image of this scene—the simulation of perception has clearly resulted in accurate perspective taking. Objects 1 and 2 are observed, but object 3 is outside the field-of-view on the table to the right. Figure 7 B shows the scene during part 2, after the demonstrator has rotated 45 degrees to the right. The observer robot realises through perspective taking that object 3 is now visible to the demonstrator, and objects 1 and 2 are not.

Table 2 shows the results from part 3 — moving object 1 while it can be seen by the observer, but not by the demonstrator. Through perceptual perspective taking the observer knows that the demonstrator cannot see the object being moved — and so, it updates its *own* cognitive map with the change in position, but not the demonstrator's. This leads to the discrepancy between the object 1 position values for the observer and the demonstrator, as shown in the table. The demonstrator *believes* that the object is in the place where it last saw it, whereas the observer *knows* it to be somewhere else.

Table 2. Cognitive map entries for observer and demonstrator after movement of object 1 inside the observer's field of view but outside of the demonstrator's. Through perceptual perspective taking, the observer knows that the demonstrator cannot see object 1 while it is being moved, and so the demonstrator's cognitive map is not updated with the changes in position and orientation as the object is moved.

Object	**X** (m)	**Y** (m)	**Z** (m)	**Angle** (Degrees)
Observer's Cognitive Map				
Object 1	0.12	0.06	0.55	20.96
Object 2	0.24	-0.26	1.82	55.07
Object 3	0.05	-0.23	2.17	40.86
Demonstrator's Cognitive Map				
Object 1	0.26	-0.03	0.83	64.40
Object 2	0.24	-0.26	1.82	55.07
Object 3	0.05	-0.23	2.17	40.86

5.2 Experiment 2

Building on the success of Experiment 1, the perspective taking was linked to the HAMMER architecture for action simulation experiments. Figure 8 shows the results of the applicability calculations for Experiment 2. Each episode is separated by a period of zero applicability, before the action simulations begin and then reset five seconds later. Figure 8 A shows the applicability levels when the observer is drawing on its representation of the demonstrator's cognitive map to generate state information for the inverse models, and in Figure 8 B the observer is using its own cognitive map. The final applicability levels achieved by each inverse model are shown in Table 3.

The top graph effectively shows the observer's attempt to determine, through simulation, what actions the demonstrator *believes* it can perform; the lower graph is the observer calculating what actions the demonstrator can actually perform, given the state of the world as the observer knows it to be. In the first three episodes, the demonstrator's cognitive map and the observer's own are in agreement as to what inverse models are applicable: the 'nudge object 3' inverse model is not simulated for the demonstrator in the first two episodes as the observer determines that the demonstrator is unaware of object 3's existence (through the perceptual perspective taking). While the demonstrator is looking at object 3, object 1 is moved to an un-nudgeable position; the demonstrator does not see this, but the observer does, the result being that the observer calculates that the demonstrator still believes that touching object 1 is possible, even though it itself knows that the action cannot be accomplished. Upon the demonstrator rotating back to observe objects 1 and 2 in episode 5, the false belief is resolved and the applicability levels are once again in agreement.

5.3 Experiment 3

Experiment 3 took the perspective taking-action simulation of Experiment 2 further, by having the observer use its PTVRs to predict the visual feedback resulting from potential demonstrator actions, and through this, predict the updates to the demonstrator's cognitive map and how this would effect the outcome of each action. Figure 9 shows the results. Figure 9 A shows the applicability levels of the three inverse models over the four episodes, as determined by the observer when observing the demonstrator and basing its action simulations on its representation of the demonstrator's cognitive map. Figure 9 B, C and D show the cognitive map updates predicted by each of the three 'ghosts', as used by the observer during prediction of visual feedback.

In this experiment, the demonstrator robot may not see an object being moved *at the time*, but if it believes an action with that object

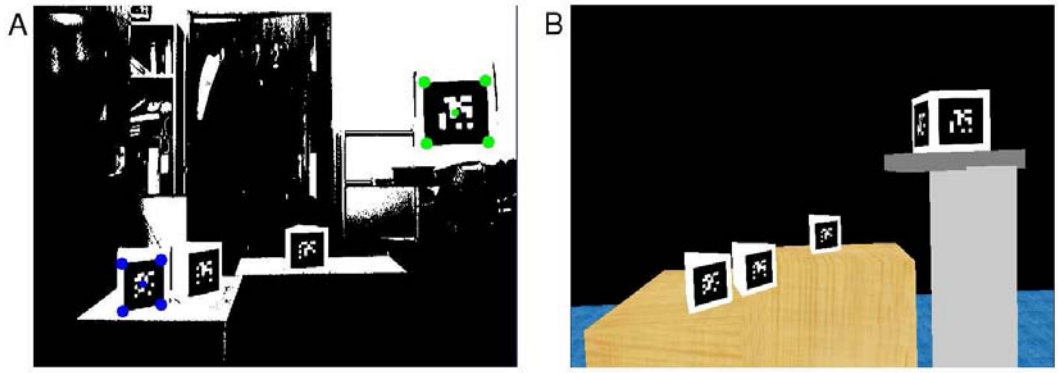


Figure 6. The observer's view of the scene. (A) shows the thresholded camera image sent to the inverse vision models. Objects 1 and 2 are on the table facing the demonstrator, and object 3 is on the table facing the observer. (B) shows the observer's cognitive map, rendered by OpenGL. The three fiducial markers can clearly be seen on the tables, and the demonstrator robot (and its 'hat') can be seen to the right.

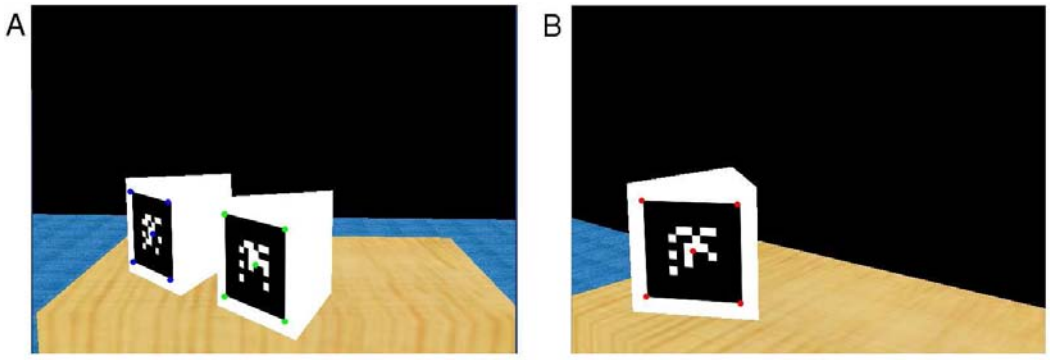


Figure 7. The demonstrator's view of the scene, re-created by the observer in simulation. (A) shows the what the demonstrator sees at the beginning of the experiment, objects 1 and 2. In (B), the demonstrator robot has rotated 45 degrees to the right, and the observer determines that it is able to see object 3. The demonstrator's actual view of (A) is shown in Figure 5.

is still possible and begins to execute it, then after it has rotated and seen the new object configuration, its cognitive map will be updated, the applicability of the action re-calculated, and then it will stop execution since it realises the action is now impossible. Episodes 3 and 4 show this; while the demonstrator is looking at object 3, objects 1 and 2 are moved away from the edge of the table. The demonstrator still believes that the objects are touchable, and so the observer sends out 'ghosts' to simulate how the action may unfold. The spikes in Figure 9 B and C show the predicted updates to the demonstrator's cognitive map when it sees that the objects have moved; as can be seen from figure Figure 9 A, negative applicabilities are calculated and the observer predicts the demonstrator will stop executing those actions. In episode four, the demonstrator rotates to observe the new configuration of objects 1 and 2, and unseen, object 3 is moved away. Figure 9 D shows the resulting cognitive map update for that episode. Again, the result is that the inverse model is no longer applicable and the action is halted mid-execution.

6 Discussion

In developmental psychology, several experimental tasks have been devised in order to investigate the development of cognitive perspective-taking abilities in the paradigm of *false belief*. One of the first tasks in this field was devised by the developmental psychologists Heinz Wimmer and Josef Perner, in response to Daniel Dennett's critique of the experimental protocols used by David Premack and Guy Woodruff in their seminal article that originated the term 'Theory of Mind' [19, 14]. This is the *action prediction* task (also known as the "unexpected transfer" task).

The action prediction task tests an observer determining what a target agent will do when holding a false belief about the world. The test subject, usually a child, observes a puppet-show involving the main character, "Maxi", and his mother. In the show, Maxi watches his mother place a chocolate bar inside a box. Maxi then leaves the room and his mother transfers the chocolate from that box into a different one. Maxi then returns, and the subject is asked where he will look for the chocolate. Further questions include what Maxi would tell to someone he wants to deceive as to the location of the chocolate, and someone he would want to tell the truth to. The result of this task is that four-year-old children give predictions based on correctly attributing the false belief, whereas younger children do not.

Through the use of the cognitive map perspective taking described in this paper, the observer would be able to solve this task. By being able to represent the cognitive map of the demonstrator robot separate to its own, the observer robot is intrinsically able to represent the concept that the demonstrator may possess a false belief about the

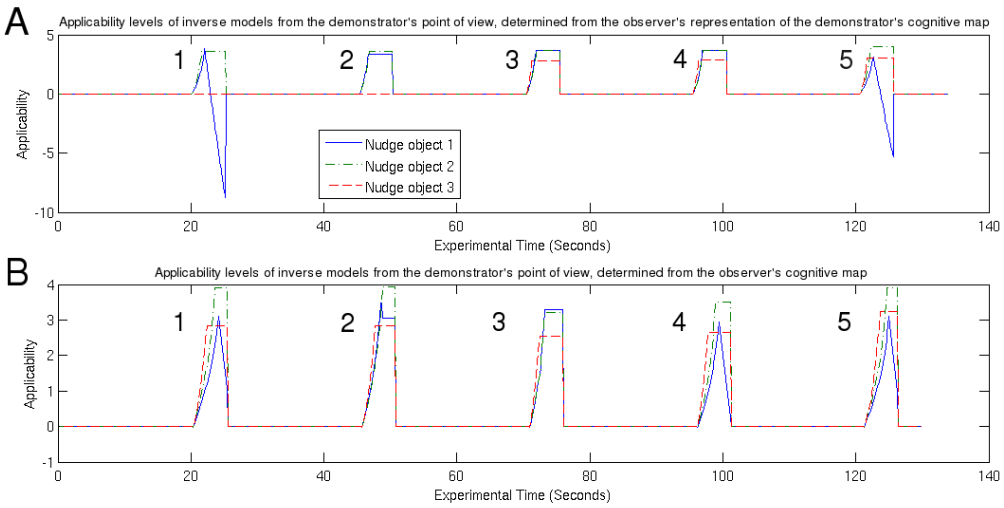


Figure 8. Applicability levels of the observer's inverse models to the demonstrator, over five repeated episodes. Each episode lasted five seconds, after which the applicability levels were reset to zero. Table 3 shows the final applicability levels for each inverse model at the end of the each episode.

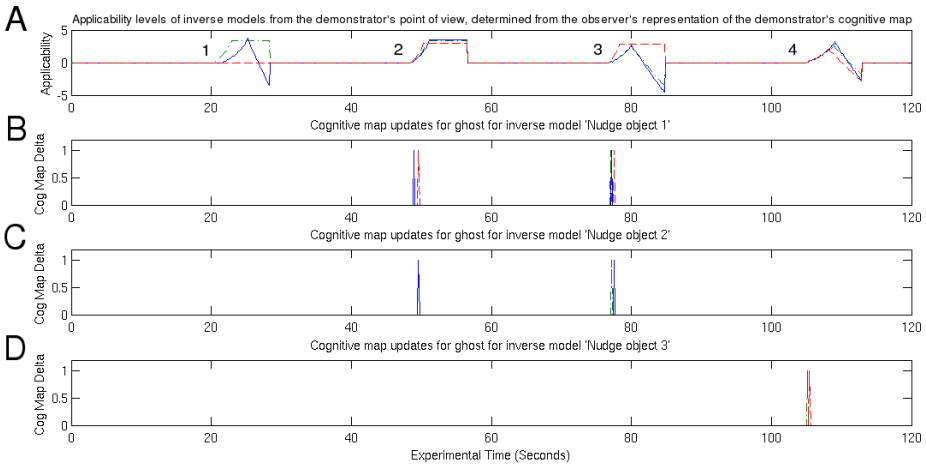


Figure 9. **A.** Applicability levels of the observer's inverse models to the demonstrator, over four repeated episodes. **B, C, D.** Cognitive map updates for each of the three 'ghosts', executing the inverse models 'nudge object' 1, 2, and 3 respectively. A spike indicates that the 'ghost' has seen something that necessitates a change to the cognitive map, and an update is made accordingly. The legend for these graphs is the same as for Figure 8.

location of objects in the world, due to objects moving outside the field-of-view, or object movement being obscured due to occlusions within the field of view. When asked to make predictions as to what the demonstrator may do in such situations, the observer robot is then able to take into account the false belief in the demonstrator's goal setting and action planning. This is illustrated through the results to part 3 of experiment 1, detailed in section 5.1. Through perceptual perspective taking the observer knows that the demonstrator cannot see object 1 being moved — and so, it updates its *own* cognitive map with the change in position, but not the demonstrator's.

Knowledge of this kind, as to the presence of false beliefs in observed agents, can be used by an observer to determine what actions a target agent considers to be available to it, as opposed to what actions it can in fact perform. This information is useful when priming a Simulation-Theory based architecture, such as the HAMMER architecture, with the action simulations it requires for action recognition. The demonstrator will derive its own action goals from what it believes to be the state of the world and move accordingly, and without a representation of the demonstrator's cognitive map, the observer will feed its perspective transform with its own world-state beliefs and potentially end up hypothesizing different goals for its action generation systems — this results in the comparison between internally generated action and observed action being meaningless. In other words, using perceptual perspective taking alone means that we see the world as *we* believe it is from the demonstrator's point of view, whereas what we need to do, in order to infer intention, is see the world as the *demonstrator* believes it is from the demonstrator's point of view. To do the former is to risk not recognising the demonstrator's movements and their action context at all, or to mis-recognise the action as being something else, or to be unable

Table 3. Final applicability levels for each inverse model shown in Figure 8. 'D' indicates that the observer is using its representation of the demonstrator's cognitive map when determining what inverse models are applicable; 'O' indicates that the observer is using its own cognitive map to determine the applicability level of the inverse models. The numbers highlighted in bold type, for 'Nudge object 1' in episode 4, indicates the situation where the observer determines that the demonstrator may possess a *false belief* as to the actions it can make. The absence applicability levels for 'Nudge object 3' in episodes 1 and 2 is due to the demonstrator robot being unaware, at that stage, of the existence of object 3.

	Episode 1		**Episode 2**		**Episode 3**		**Episode 4**		**Episode 5**	
I-Model	D	O	D	O	D	O	D	O	D	O
Nudge object 1	-8.80	1.07	3.35	3.05	3.65	3.28	**3.65**	**0.26**	-5.28	0.98
Nudge object 2	3.59	3.89	3.59	3.93	3.69	3.22	3.69	3.50	3.97	3.90
Nudge object 3	—	2.85	—	2.86	2.83	2.55	2.87	2.65	3.01	3.25

to interpret the demonstrator's goal, and therefore be unable to imitate or learn. The results for experiments 2 and 3 show how through coupling the perspective taking architecture developed in this paper to the action simulation capabilities of HAMMER, the observer can successfully *predict* and *attribute* actions to the demonstrator, while taking into account prior knowledge and experience, and potential false beliefs.

In previous research, the HAMMER architecture was used to model and make predictions regarding the visuomotor 'mirror' neurons found in area F5 of macaque monkey premotor cortex [3]. These neurons are active both when observing an object-directed action, and when performing the same action, leading to suggestions that they underly the imitation capability. Recently, it was found that a subset of these neurons fire even when the object goal of the action is hidden from view, so long as the observer has prior knowledge of the object's presence [17]. With the addition of the cognitive map mechanism described in this paper, HAMMER gains this capability, by keeping a long-term memory of the locations of objects. This can be seen in the results for episode 2 of Experiment 2, where object 1 occludes object 2 from the observer's sight, but the action simulation is still performed. Furthermore, the results of section 5 offer a further prediction—that when a demonstrator performs an action based on a known *false belief* as to the presence of an object, the observer's mirror neurons will fire. Although there is currently no evidence either way, this would lend support to the hypothesis that the mirror neurons encode *intention* and underly action understanding, in addition to action recognition.

7 Conclusions

In this paper we have presented a perspective-taking architecture that uses simulation of visual perception to build up and maintain representations of the cognitive map of a demonstrator. This mechanism, used to improve the state information provided to the HAMMER imitation architecture, was deployed onto robots for perspective-taking and action-prediction experiments, in which an observer successfully attributed potential actions and action predictions to a demonstrator possessing false beliefs regarding the environment. In future work, the mechanism will extended and investigated in experiments involving the observer inferring false beliefs from the actions of a demonstrator.

REFERENCES

[1] E. K. Ackermann, 'Perspective-taking and object construction: Two keys to learning', in *Constructionism in Practice: Designing, Thinking and Learning in a Digital World*, eds., Y. Kafai and M. Resnick, chapter 2, 25–37, Mahwah, New Jersey: Lawrence Erlbaum Associates, first edn., (1996).

[2] M. Billinghurst, H. Kato, and I. Poupyrev, 'The magicbook: A transitional AR interface', *Computers and Graphics*, 745–753, (November 2001).

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

[4] Y. Demiris and M. Johnson, 'Distributed, predictive perception of actions: A biologically inspired robotics architecture for imitation and learning', *Connection Science*, **15**(4), 231–243, (December 2003).

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

[6] A. Drolet, R. Larrick, and M. W. Morris, 'Thinking of others: How perspective taking changes negotiators' aspirations and fairness perceptions as a function of negotiator relationships', *Basic and Applied Social Psychology*, **20**(1), 23–31, (1998).

[7] A. D. Galinsky, G. Ku, and C. S. Wang, 'Perspective-taking and self-other overlap: Fostering social bonds and facilitating social coordination', *Group Processes and Intergroup Relations*, **8**(2), 109–124, (2005).

[8] V. Gallese, 'The manifold nature of interpersonal relations: the quest for a common mechanism', *Phil. Trans. of the Royal Society of London B*, **358**, 517–528, (2003).

[9] R. M. Gordon, 'Simulation vs theory-theory', in *The MIT Encyclopædia of the Cognitive Sciences*, eds., R. A. Wilson and F. Keil, 765–766, MIT Press, (1999).

[10] M. Johnson and Y. Demiris, 'Hierarchies of coupled inverse and forward models for abstraction in robot action planning, recognition and imitation', in *Proceedings of the AISB 2005 "Third International Symposium on Imitation in Animals and Artifacts"*, (April 2005).

[11] M. Johnson and Y. Demiris, 'Perceptual perspective taking and action recognition', *International Journal of Advanced Robotic Systems*, **2**(4), 301–308, (December 2005).

[12] K. S. Narendra and J. Balakrishnan, 'Adaptive control using multiple models', *IEEE Transactions on Automatic Control*, **42**(2), 171–187, (February 1997).

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

[14] D. Premack and G. Woodruff, 'Does the chimpanzee have a theory of mind?', *Behavioral and Brain Sciences*, **4**, 515–526, (1978).

[15] A. C. Schultz and J. G. Trafton, 'Towards collaboration with robots in shared space: spatial perspective and frames of reference', *Interactions*, 22–23, (March-April 2005).

[16] G. Simmons and Y. Demiris, 'Object grasping using the minimum variance model', *Biological Cybernetics*, **94**(5), 393–407, (May 2006).

[17] M. A. Umilta, E. Kohler, V. Gallese, L. Fogassi, L. Fadiga, C. Keysers, and G. Rizzolatti, 'I know what you are doing: A neurophysiological study', *Neuron*, **31**, 155–165, (July 2001).

[18] Y. Wada and M. Kawato, 'A neural network model for arm trajectory formation using forward and inverse dynamics models', *Neural Networks*, **6**, 919–932, (1993).

[19] H. Wimmer and J. Perner, 'Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception', *Cognition*, **13**, 103–128, (1983).

[20] D. M. Wolpert, K. Doya, and M. Kawato, 'A unifying computational framework for motor control and social interaction', *Phil. Trans. of the Royal Society of London B*, **358**, 593–602, (2003).

[21] D. M. Wolpert and M. Kawato, 'Multiple paired forward and inverse models for motor control', *Neural Networks*, **11**, 1317–1329, (1998).

Learning by Observation: Comparison of Three Methods of Embedding Mentor's Knowledge in Reinforcement Learning Algorithms

Natalia Akchurina[1]

Abstract. Using knowledge of already successfully functioning agents can help to avoid expensive exploration that is so vital to some domains of reinforcement learning. Three methods to embed mentor's knowledge are proposed: initialization of Q function, reward shaping and implementing mentor's decisions in a separate action-value function. The speed of convergence of these methods in combination with Q-learning algorithm with different amount of information on mentor's decisions and their robustness to the quality of mentor are compared on four domains from the benchmarks for testing and comparing reinforcement learning algorithms "Reinforcement Learning Benchmarks and Bake-offs".

1 INTRODUCTION

Reinforcement learning has attracted rapidly increasing interest in the machine learning and artificial intelligence communities in the last years. Its promise is very tempting — a way of programming agents by reward and punishment without specifying how the task is to be achieved. The framework permits the simplifications necessary in order to make progress in domains that are clearly beyond our current capabilities due to many unsolved key problems in learning and representation.

Reinforcement learning is the problem faced by an agent that must learn to choose optimal sequences of actions through trial-and-error interactions with its environment. This very general problem covers tasks such as learning to control a mobile robot (robot control problems, such as navigation, pole-balancing, or juggling are the canonical reinforcement-learning problems), learning to optimize operations, learning to play board games (TD-Gammon program [16], based on reinforcement learning, has become a world-class backgammon player) or set prices on virtual markets [2]. The task of the agent is to learn from indirect, delayed reward a policy that maps states of the world to the actions the agent should take to maximize its cumulative reward. Trial-and-error search and delayed reward — are the two most important distinguishing features of reinforcement learning. One of the most important breakthroughs in reinforcement learning was the development of Q-learning algorithm. Q-learning can acquire optimal policies from delayed rewards, even when the agent has no prior knowledge of the effects of its actions on the environment.

While learning the agent must choose whether to explore unknown states and actions (to gather new information), or hold to states and actions that it has already learned will yield high rewards (to maximize its cumulative reward).

For the agent learns through trial and error the quality of its decisions is rather poor at the beginning. Some mistakes in exploration can even lead to vital consequences. For example, while learning to drive a bicycle by reinforcement [12] the robot fell so much and got deformed so badly that could not be used anymore for any purpose let alone to balance on the bicycle. On virtual markets a real profit can be wasted while learning how to set the prices [2, 17].

How can this expensive exploration be avoided?

In many environments already exist agents that function successfully and whose knowledge can be of use to avoid committing vital mistakes and making expensive exploration.

But how can we tell the difference between "experts" and those agents whose policies are not so effective? In general case we don't know in advance how good the agent's decisions are.

To implement agent's experience under this condition we need new methods that must possess the following properties:

- They must be robust to the quality of the expert (they must learn the optimal policy even if the expert's decisions turned out to be not so good)
- Usage of bad decisions should not radically worsen the speed of the convergence of the methods
- Usage of "real expert's" decisions should improve the convergence of the methods even if the data were scarce

Evident solution seems to embed somehow experience in existing good reinforcement learning algorithm.

In this paper we are proposing three methods to implement expert's knowledge in Q-learning algorithm (though they can be used with any action-value function based reinforcement learning algorithms): initialization of Q function, reward shaping and implementing mentor's decisions in a separate action-value function. We suppose that expert's knowledge is available in the form of state→action pairs (what action is better to take in this particular state) but this state-action pairs don't constitute the whole policy but come as a result of observing the expert agent for some time (to obtain such state-action pairs by observation is quite possible for many environments [14, 17, 5, 6]). This is a much weaker assumption than the supposition that the agent can get hold of state→action→next state→reward sequences [9, 7, 8, 15, 18, 19]. It also allows us not only to avoid problems with communication but to learn from a broader range of agents including those who just don't care to share their experience with us as well as competitive agents who in no way concern them-

[1] International Graduate School of Dynamic Intelligent Systems, University of Paderborn, Germany, email: anatalia@mail.uni-paderborn.de

selves with our successful learning[2].

The problem of learning a control policy to maximize cumulative reward is in general one of learning to control sequential processes. Subsection 2.1 is devoted to the introduction of the general formulation of this problem based on Markov decision process. In subsection 2.2 we present Q-learning algorithm. Three methods to embed expert's knowledge in action-value reinforcement learning algorithms are given in section 3. Section 4 is devoted to the depiction of the domains on which the three methods were tested and section 5 — to the results of the experiments. The discussion how well the proposed methods satisfy the above formulated requirements is presented in section 6. In a few words, reward shaping turned out to be quite inapplicable for the above problem. Q-initialization has proven to be robust to mentor's level of expertise, meanwhile the last method requires a very small amount of data on mentor's decisions.

2 PRELIMINARIES

2.1 Definitions

Here we introduce a general formulation of the problem of learning sequential control strategies, based on Markov decision process.

In a Markov decision process (MDP) [11] the agent can perceive a set S of distinct states of its environment and has a set A of actions that it can perform. At each discrete time step t, the agent senses the current state s_t, chooses a current action a_t, and performs it. The environment responds by giving the agent a reward $r_t = r(s_t, a_t)$ and by producing the succeeding state s_{t+1} with the transition probability $P_{s_t a_t}(s_{t+1})$. Here the function r and the transition probabilities $P_{sa}(\cdot)$ are part of the environment and are not necessarily known to the agent. In an MDP, r and $P_{sa}(\cdot)$ depend only on the current state and action, and not on earlier states or actions. We consider only the case in which S and A are finite.

Formally, a *finite nondeterministic MDP* is a tuple $M = (S, A, \delta, \gamma, r)$, where

- S is a finite set of m *states*
- $A = \{a_1, a_2, \ldots, a_n\}$ is a set of n *actions*
- $\delta = \{P_{sa}(\cdot) | s \in S, a \in A\}$, where $P_{sa}(\cdot)$ are the *state transition probabilities* upon taking action a in state s
- $\gamma \in [0, 1)$ is the *discount factor*
- $r : S \times A \to \mathbb{R}$ is the *reward function*, bounded in absolute value by r_{max}

A *policy* is a function $\pi : S \to A$, that maps the current observed state s_t to action a_t, $\pi(s_t) = a_t$. The expected value $V^\pi(s_t)$ of the *discounted cumulative reward* achieved by following an arbitrary policy π from an arbitrary initial state s_t is defined as follows:

$$V^\pi(s_t) \equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots] \equiv E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]$$

where the sequence of rewards r_{t+i} is generated by beginning at state s_t and by repeatedly using the policy π to select actions as described above (i.e., $a_t = \pi(s_t)$, $a_{t+1} = \pi(s_{t+1})$, etc.). Here $0 \leq \gamma < 1$ is the *discount factor* that determines the relative value of delayed versus immediate rewards.

[2] For the sake of consistency, we will use the terms "mentor" and "expert" to describe any agent from which we can learn by observation even if the mentor (expert) is hostile and we only suspect it to possess enough level of expertise to function effectively in the environment

Agent's learning task is to learn a policy π that maximizes $V^\pi(s)$ for all states s. We will call such a policy an *optimal policy* and denote it by $\pi^\star$.

$$\pi^\star \equiv arg\max_\pi V^\pi(s), (\forall s)$$

$V^{\pi^\star}(s)$ gives the maximum expected value of the discounted cumulative reward that the agent can obtain starting from state s; that is, the expected value of the discounted cumulative reward obtained by following the optimal policy beginning at state s.

The optimal action in state s is the action a that maximizes the expected value of the sum of the immediate reward $r(s, a)$ plus the value $V^{\pi^\star}$ of the immediate successor state, discounted by γ.

$$\pi^\star(s) \equiv arg\max_a E_{s' \sim P_{sa}(\cdot)}[r(s, a) + \gamma V^{\pi^\star}(s')]$$

(where the notation $s' \sim P_{sa}(\cdot)$ means that s' is drawn according to the distribution $P_{sa}(\cdot)$)

Let us define the evaluation function $Q(s, a)$ so that its value is the maximum expected value of the discounted cumulative reward that can be achieved starting from state s and applying action a as the first action. In other words, the value of Q is the expected value of the reward received immediately upon executing action a from state s, plus the value (discounted by γ) of following the optimal policy thereafter.

$$Q(s, a) \equiv E_{s' \sim P_{sa}(\cdot)}[r(s, a) + \gamma V^{\pi^\star}(s')]$$

Therefore, optimal policy in terms of $Q(s, a)$ will be

$$\pi^\star(s) = arg\max_a Q(s, a)$$

2.2 Q-learning

One of the most important breakthroughs in reinforcement learning was the development of Q-learning algorithm. Q-learning can acquire optimal control policies from delayed rewards, even when the reward function and / or state transition probabilities are not known. All that is required for correct convergence is that the system can be modeled as a nondeterministic Markov decision process, the reward function r is bounded, and actions are chosen so that every state-action pair is visited infinitely often.

Q-learning algorithm for episodic tasks (with terminal states):
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):

- Initialize s
- Repeat (for each step of episode):
  - Choose a from s using policy derived from Q (e.g., ϵ-greedy)
  - Take action a, observe r, s'
  - $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
  - $s \leftarrow s'$;
- until s is terminal

Parameter α in Q-learning algorithm must satisfy $0 < \alpha \leq 1$.

Choosing a from s using ϵ-greedy policy derived from Q means that most of the time we choose an action a that has maximal estimated action value $Q(s, a)$, but with probability ϵ we instead select an action at random.

3 SUPERVISED REINFORCEMENT LEARNING METHODS

Here we are proposing three methods to embed expert's decisions in Q-learning algorithm.

We suppose that state-action pairs (s, a) of expert's choice are available, that is not such bold an assumption for many environments [14, 17, 5, 6].

3.1 Q-initialization

Q-initialization was before mentioned in [14] to use prior knowledge to accelerate learning.

To embed the expert's knowledge in Q-learning algorithm we can use it thus:

Let's consider state s.

Let $\{a_1, a_2, \ldots, a_n\}$ be the set of the possible actions in this state s.

If the expert agent has chosen action a_k, $1 \leq k \leq n$, at this state it means in the terms of Q-function that:

$$\begin{gathered} Q(s, a_k) > Q(s, a_1) \\ Q(s, a_k) > Q(s, a_2) \\ \vdots \\ Q(s, a_k) > Q(s, a_{k-1}) \\ Q(s, a_k) > Q(s, a_{k+1}) \\ \vdots \\ Q(s, a_k) > Q(s, a_n) \end{gathered}$$

By initializing $Q(s, a_k)$ to some nonzero value and $Q(s, a_i)$ to zero, $i = 1, \ldots, n$, $i \neq k$, we can take into account the expert's decision.

3.2 Reward shaping

Reward shaping is a technique of changing rewards to accelerate learning and turned out to be very useful and even vital for the following tasks:

- A foraging task of a group of mobile robots [10]
- Learning to drive a bicycle [12]

The idea of this promising approach [10, 12], which is borrowed from behavioral psychology, is to give the learning agent a series of relatively easy problems building up to the harder problem of ultimate interest. The term was first proposed in [13], who studied the effect on animals, especially pigeons and rats. To train an animal to produce a certain behavior, the trainer must find out what subtasks form the desired behavior, and how these should be reinforced. By reward shaping horses can be brought to recognize numbers and pigs to eat at a table [4].

To use reward shaping for our task we need two reward functions instead of one:

- τ — artificial reward function
- r — usual reward function from the environment

Let's consider state s.

Let $\{a_1, a_2, \ldots, a_n\}$ be the set of the possible actions in the state s.

If the expert agent has chosen action a_k, $1 \leq k \leq n$, at this state, then we initialize $\tau(s, a_k)$ with some small positive value, and $\tau(s, a_i) = 0$, $i = 1, \ldots, n$, $i \neq k$.

The idea is to bias the action choice to the expert's one.

3.3 Two action-value functions

The idea to use two action-value functions was first proposed in [18]. But there the second action-value function was used to implement immediate feedback on agent's performance provided by external critic. Here we use two action-value function to implement the idea always to trust the expert's choice and to explore states in background mode and to use its results only in case of the lack of information about the expert's decision.

Here we derive policy from the sum of two action-value functions $Q + B$:

- B — action-value function reflecting the expert's choice
- Q — usual action-value function

Let's consider state s.

Let $\{a_1, a_2, \ldots, a_n\}$ be the set of the possible actions in the state s.

If the expert agent has chosen action a_k, $1 \leq k \leq n$, at this state, then we initialize $B(s, a_k)$ so that $B(s, a_k) > r_{max}$, and $B(s, a_i) = 0$, $i = 1, \ldots, n$, $i \neq k$.

4 BENCHMARKS

We have tested the above proposed methods on four domains from the benchmarks for testing and comparing reinforcement learning algorithms "Reinforcement Learning Benchmarks and Bake-offs" [1] that were the result of two workshops "NIPS Workshop on Reinforcement Learning: Benchmarks and Bake-offs" 2004 and 2005.

4.1 Gridworld with mines

The figure 1 shows a standard gridworld, with start and goal states, but with one difference: there are mines located at various positions on the grid. These mines are stationary and cause the agent to be destroyed if touched.

- *State space*: The state is represented by an integer value of the state label, comprised of the row index (x) multiplied by the number of columns plus the column index (y), resulting in 108 unique states
- *Starting states*: Agent is started in a new random starting state at the beginning of each episode
- *Terminal states*: Mines or goal state
- *Actions*: The actions are the standard four: *up*, *down*, *right* and *left*
- *Rewards*: $+10$ for reaching the goal, -10 for hitting a mine and -1 otherwise

4.2 Distributed sensor network (DSN)

The distributed sensor network (DSN) problem is a sequential decision making variant of the distributed constraint optimization problem described in [3].

The network consists of two parallel chains of an arbitrary, but equal, number of sensors. The area between the sensors is divided into cells. Each cell is surrounded by exactly four sensors and can be occupied by a target. With equal probability a target moves to the cell to its left, to the cell to its right or remains on its current position. Actions that move a target to an already occupied cell are not executed. It is the goal of the sensors to capture all targets. See figure 2 for a configuration with eight sensors and two targets.

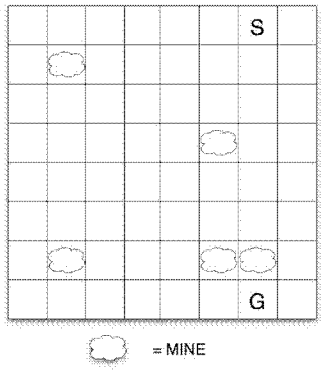


Figure 1. Gridworld with mines

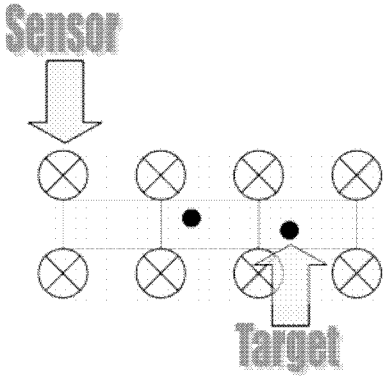


Figure 2. Sensor network configuration with eight sensors and two targets

Each sensor is able to perform three actions: track a target in the cell to its immediate left, to its immediate right, or don't track at all. Every track action has a small cost (reward of -1). When in one time step at least three of the four surrounding sensors track a target, it is 'hit'. Each target starts with a default energy level of three. Each time a target is hit its energy level is decreased by one. When it reaches zero the target is captured and removed. The three sensors involved in the capture are each provided with a reward of $+10$. An episode finishes when all targets are captured. Each joint action comprises single actions and the received reward is the sum of the individual agent rewards.

- *State space* (37): 9 states for each of the 3 configurations with 2 agents, 9 for those with one agent and 1 for those without any agents
- *Starting states* (3): $[3, 3, 0]$, $[3, 0, 3]$, $[0, 3, 3]$ (where the starting state $[3, 3, 0]$ corresponds to the initial situation when the two targets are in the first two cells and possess the maximum energy level 3)
- *Terminal states* (1): when both targets have zero hit points $[0, 0, 0]$
- *Actions* (6561): the actions of each sensor (0—don't track, 1—track left cell, 2—track right cell) are packed into a single integer a; the ith sensor's action is: $\left(\frac{a}{3^{7-i}}\right) mod 3$
- *Rewards* ($[-8, 54]$): -1 for each sensor focus, $+30$ for eliminating a target

This problem has a relatively small state space compared to the action space. Furthermore, the problem involves multiple targets forcing the sensors to coordinate their actions.

4.3 Cat and mouse

There is a cat, a mouse, a piece of cheese as well as some obstacles in the cat and mouse world. The mouse tries to avoid getting caught by the cat, at the same time trying to get to the cheese. The mouse is the one learning, the cat is already programmed to go for the mouse. The obstacle layout is generated randomly. Both the cat and mouse have 8 degrees of movement: up, down, left and right, as well as the four diagonals. The mouse gets positive reward ($+50$) for getting the cheese. The mouse gets the cheese when it is in the same square as the cheese. The mouse gets negative reward (-100) for getting caught, by simply moving to the same square as the cat. The episode ends when the cat catches the mouse. Single cat and piece of cheese are in the maze at one time.

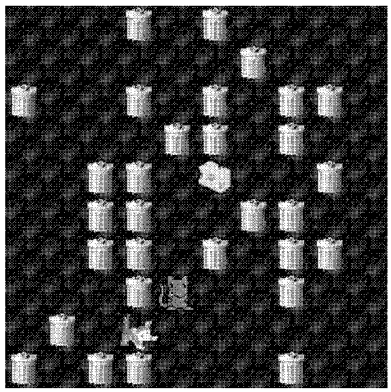


Figure 3. Cat and mouse game [1]

- *State space*: mouse position, cat position, cheese position
- *Starting states*: Mouse and cat start in a new random starting state at the beginning of each episode
- *Terminal states*: The episode ends when the cat catches the mouse
- *Actions*: Mouse has 8 degrees of movement: up, down, left and right, as well as the four diagonals
- *Rewards*: The mouse gets positive reward $(+50)$ for getting the cheese. The mouse gets negative reward (-100) for getting caught

4.4 Blackjack

The problem is based on the Blackjack problem described in Sutton-Barto's book [14]. Standard rules of blackjack hold, but modified to allow players to (mistakenly) hit on 21. Episodes are prevented from starting in terminal states.

- *State space*: 1-dimensional integer array of 3 elements:
  - element[0] – current value of player's hand (4-21)
  - element[1] – value of dealer's face-up card (2-11)
  - element[2] – player does not have usable ace (0/1)
- Region for *starting states*: player has any 2 cards (uniformly distributed), dealer has any 1 card (uniformly distributed)
- *Terminal states*: Terminates when player "sticks" or is "bust"
- *Actions*: discrete integers 0-HIT, 1-STICK
- *Rewards*: -1 for a loss, 0 for a draw and 1 for a win

5 EXPERIMENTS

The above considered domains were used for testing the speed of convergence of the proposed methods in combination with Q-learning algorithm with different amount of information about mentor's decisions and their robustness to the quality of the mentor.

As we want to avoid expensive exploration we have also no chance to tune parameters and so we used the same parameters for all the domains: $\gamma = 0.9$, $\alpha = 0.1$ and $\epsilon = 0.1$.

We considered the rate of reduction in exploration on the basis of cumulative reward (the more cumulative reward is — the better reduction has been achieved) on episodes necessary for the methods to converge (10000 episodes were necessary for cat and mouse problem, and 1000 for the others). As mentor we chose the agent trained with the use of Q-learning algorithm for all the domains except blackjack (Q-learning algorithm failed to converge to optimal blackjack strategy [14] with the above parameters in 1000 episodes, so we used optimal blackjack player [14] as the mentor). We also initialized the corresponding functions so that they reflected random action choice in every state to research the robustness of our methods in case of a "poor mentor". For in general we don't possess information even about the order of the rewards in the environment, we used 0.1 as the initial value for functions in Q-initialization and reward shaping methods.

For gridworld with mines, DSN and cat and mouse problems the following comparison on the cumulative reward was held for every method proposed in subsections 3.1, 3.2 and 3.3:

- Q-learning — Q-learning algorithm presented in subsection 2.2
- 5 mentor episodes — current method with data on the mentor's state $\rightarrow$ action choice during 5 episodes
- 10 mentor episodes — current method with data on the mentor's state $\rightarrow$ action choice during 10 episodes
- 100 mentor episodes — current method with data on the mentor's state $\rightarrow$ action choice during 100 episodes
- 1000 mentor episodes — current method with data on the mentor's state $\rightarrow$ action choice during 1000 episodes
- Randomly initialized — the corresponding function was initialized so that it reflected random action choice in every state (to research the robustness of the current methods in case of a "poor mentor")

For blackjack the following comparison on the cumulative reward was held for every proposed method:

- Optimal policy — the optimal policy known for blackjack game [14] (player with this strategy was used as the mentor)
- Q-learning — Q-learning algorithm presented in subsection 2.2
- 5 mentor episodes — current method with data on the optimal player's state $\rightarrow$ action choice during 5 games
- 10 mentor episodes — current method with data on the optimal player's state $\rightarrow$ action choice during 10 games
- 100 mentor episodes — current method with data on the optimal player's state $\rightarrow$ action choice during 100 games
- 1000 mentor episodes — current method with data on the optimal player's state $\rightarrow$ action choice during 1000 games
- Randomly initialized — the corresponding function was initialized so that it reflected random action choice in every state (to research the robustness of the current methods in case of a "poor mentor")

As we consider the rate of reduction in exploration on the basis of cumulative reward on episodes necessary for the methods to converge the slope on such graphics corresponds to the quality of the learnt policy and the initial low values of cumulative reward reflect the errors in exploration (the initial bad quality of agent's decisions).

5.1 Gridworld with mines

- As we can see on figure 4 the agent with original Q-learning algorithm (without exploiting any expert knowledge) learnt good policy approximately after 200 episodes for gridworld with mines benchmark. Q-initialization method with any (even random) initialization acquired good policy also approximately after 200 episodes (the slopes of the graphics became the same as for Q-learning method). Even the usage of 5 mentor's episodes allowed to reduce the errors in exploration in the beginning but the agent stopped committing mistakes only when 100 mentor's episodes became available. And the more expert's decisions it used the better reduction in exploration it got (the larger the cumulative reward on figure 4). Agent with randomly initialized Q function at first committed many mistakes but nevertheless after 200 episodes it learnt a good policy.
- The results of testing reward shaping method on gridworld with mines domain is presented on the figure 5. As we can see only the usage of 1000 mentor episodes is comparable with initial Q-learning (these two graphics coincide on the figure). Embedding mentor's decisions in Q-learning algorithm with the use of reward shaping worsened the convergence of the original method. And the problem is not just with the exploration phase (the time till the agent learnt a good policy — corresponds to the point on the graphic where the cumulative reward starts to grow) but the agent also needed more episodes to learn it (the growth of corresponding cumulative reward curves comes later). Though as we can see on the figure 5 reward shaping with any initialization allowed to learn a good policy (at the end the slopes of the graphics are the same as for Q-learning method).

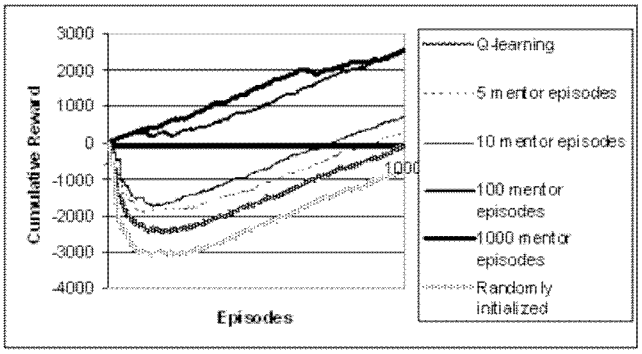


Figure 4. Gridworld with mines: cumulative reward of Q-initialization

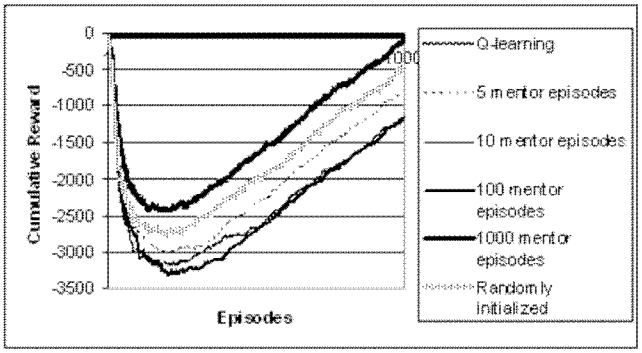


Figure 5. Gridworld with mines: cumulative reward of reward shaping

- As it can be seen on figure 6 usage of even a small number of expert's decisions (even 5 episodes) with two action-value function method prevented the agent from making mistakes. And the agent began to accumulate rewards from the very beginning. But as it can also be seen from figure 6 two action-value function method didn't show the robustness to the quality of mentor's decisions (the corresponding graphic of cumulative reward in case of randomly initialized value function goes inevitably down). So actually the agent that embedded decisions of poor agent with the help of Q-learning with two action-value functions couldn't learn a good policy at all. From figure 6 we can draw a conclusion that the more expert decisions we used the better reduction in exploration we got (the corresponding graphics of cumulative reward are higher).

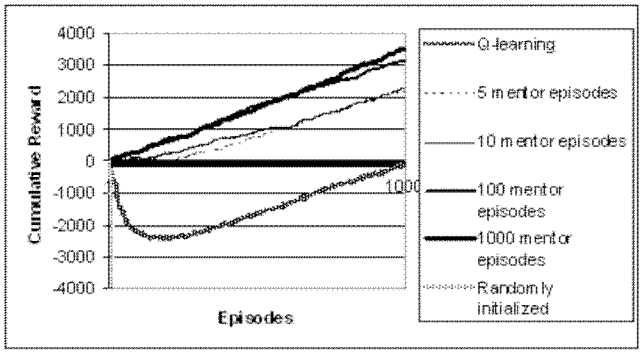


Figure 6. Gridworld with mines: cumulative reward of two functions

5.2 Distributed sensor network (DSN)

- As it can be seen from the figure 7 the more expert decisions we use for the Q-initialization method the better reduction in exploration we get for DSN domain.

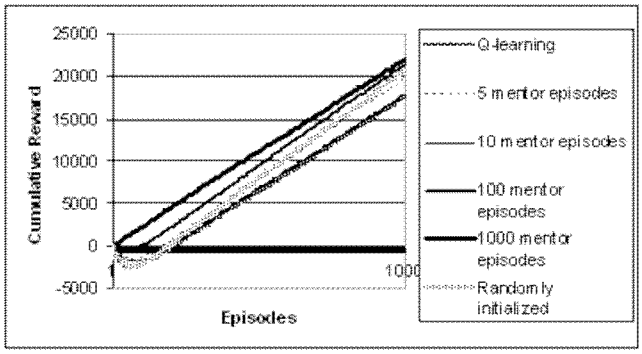


Figure 7. DSN: cumulative reward of Q-initialization

- The usage of any quantity of expert's decisions (see figure 8) didn't influence the convergence of reward shaping method for DSN benchmark.
- As it can be seen from figure 9 the usage of even a small number of expert's decisions with two action-value function method prevented the agent from making mistakes. But in case the mentor

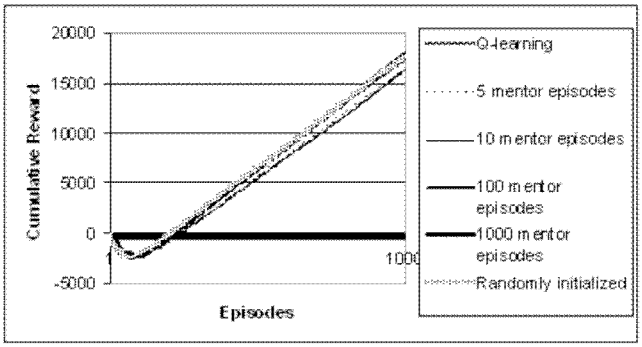


Figure 8. DSN: cumulative reward of reward shaping

itself doesn't know how to function successfully in the environment the consequences are devastating. As it ensues from the corresponding graphic two function method didn't allow the agent in this case to learn a good policy.

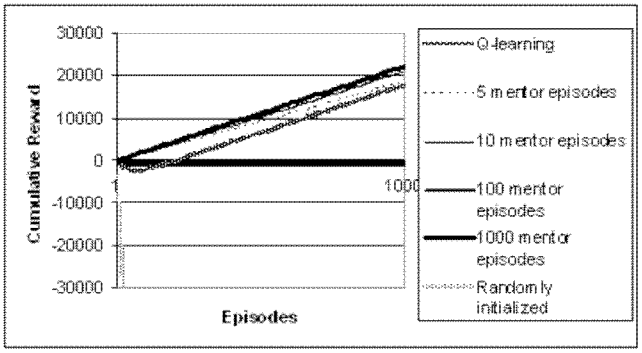


Figure 9. DSN: cumulative reward of two functions

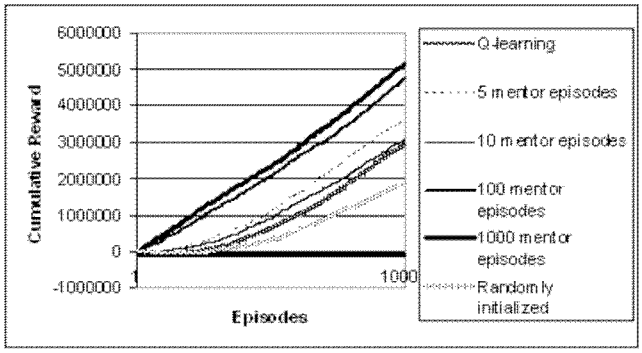


Figure 10. Cat and mouse: cumulative reward of Q-initialization

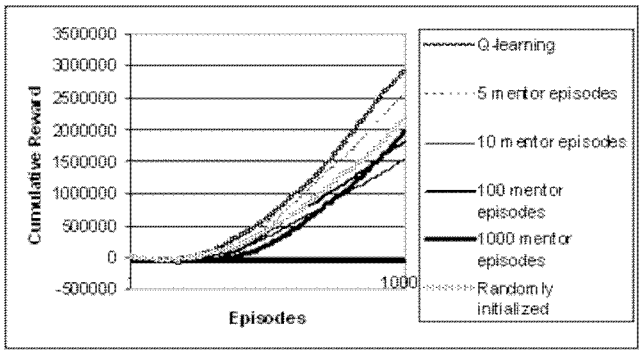


Figure 11. Cat and mouse: cumulative reward of reward shaping

5.3 Cat and mouse

- The figure 10 represents the results of testing Q-initialization method on cat and mouse benchmark. As we can see approximately after 5000 episodes Q-initialization as well as original Q-learning method allowed to learn a good policy (the slopes of the corresponding graphics after 5000 episodes are approximately the same). But usage of good mentor's decisions allowed to collect larger cumulative reward and in general the more mentor's episodes the agent observed the larger cumulative reward it got. In case of randomly initialized Q-learning function the cumulative reward is smaller then without any initialization at all (original Q-learning method).
- As it can be seen from figure 11 embedding mentor's decisions with the help of reward shaping method only worsened original Q-learning.
- We can draw a conclusion from figure 12 that the more data on mentor's decisions we used for two action-value function method the earlier the agent started to function effectively in cat and mouse benchmark. In case of a poor mentor the agent using this method had no chance to learn a good policy.

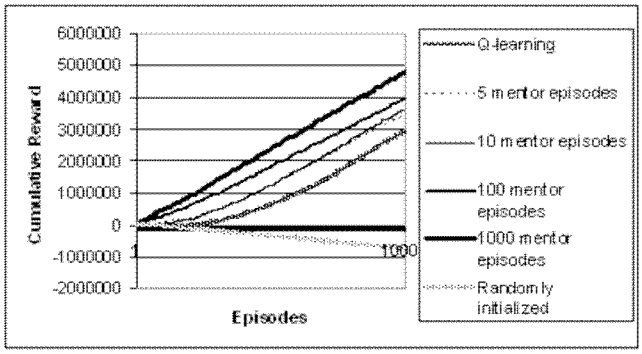


Figure 12. Cat and mouse: cumulative reward of two functions

5.4 Blackjack

When the agent can use decisions the optimal player took during 1000 games it learns optimal strategy either by Q-initialization or with two function method (see figures 13 and 15). On observing less number of games it can function only as well as Q-learning player.

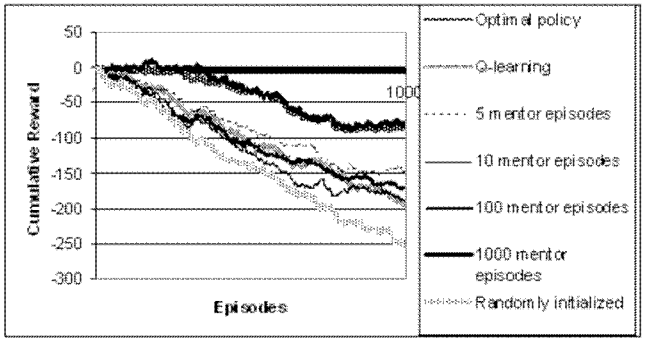


Figure 13. Blackjack: cumulative reward of Q-initialization

As in the above considered domains reward shaping method didn't yield any positive results for blackjack (see figure 14). Original Q-learning turned out to be better than reward shaping with any initialization.

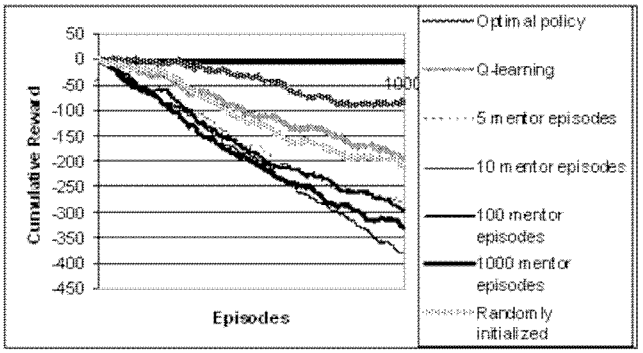


Figure 14. Blackjack: cumulative reward of reward shaping

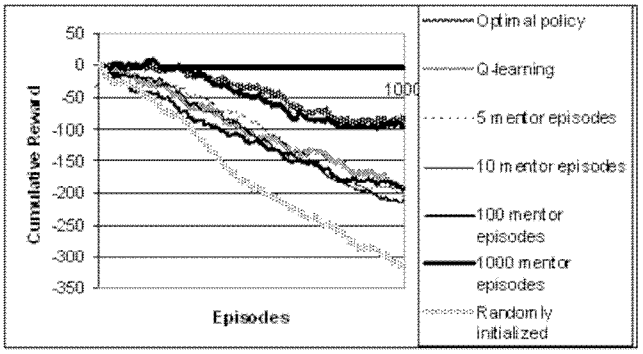


Figure 15. Blackjack: cumulative reward of two functions

6 DISCUSSION

Q-initialization and two action-value function methods proved to be useful for exploration reduction problem. The effect in reduction in case of Q-initialization got visible starting from 100 mentor episodes, while two function method started to show good results already with 5 available episodes. Meanwhile only Q-initialization showed good robustness to the quality of the mentor (that quite corresponds with theoretical results for Q-initialization method is the only one considered that is bound to converge to optimal policy for initial Q-learning algorithm converges to optimal strategy in case every state-action pair is visited infinitely often for any initial values of Q function). Reward shaping turned out to be too capricious (we tried it with different initialization values as well) but nothing yielded any good stable results. This though is quite in accordance with the first unsuccessful attempts of tuning extra reward function in reward shaping later successful applications [10, 12]. But our task differs from the one that was solved in [10, 12]. We are not interested in the solution however good it is if it requires tuning (tuning means a lot of exploration). What is clear is that reward shaping cant be seriously considered as general method for embedding expert's knowledge in reinforcement learning algorithms.

7 CONCLUSION

In this paper three methods: initialization of Q function, reward shaping and implementing mentor's decisions in a separate action-value function were proposed for an actual problem of reducing expensive exploration by means of using knowledge of already successfully functioning agent. Testing of these three methods in combination with Q-learning algorithm on four domains from the benchmarks for testing and comparing reinforcement learning algorithms "Reinforcement Learning Benchmarks and Bake-offs" has shown that reward shaping even in the best case requires too much tuning (that itself does need a lot of exploration) and quite inapplicable for the above problem. Q-initialization has proven to be robust to mentor's level of expertise, meanwhile the last method has shown better results in exploration reduction and requires less data on mentor's decisions.

ACKNOWLEDGEMENTS

We would like to thank the referees for their comments which helped improve this paper.

REFERENCES

[1] http://rlai.cs.ualberta.ca/rlbb/top.html.
[2] Natalia Akchurina and Hans Kleine Büning, 'Virtual markets: Q-learning sellers with simple state representation', *Proceedings of the Workshop on Autonomous Intelligent Systems: Agents and Data Mining (AIS-ADM 07) (to appear). Lecture Notes in Computer Science*, (2007).
[3] Syed Ali, Sven Koenig, and Milind Tambe, 'Preprocessing techniques for accelerating the dcop algorithm adopt', in *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 1041–1048, New York, NY, USA, (2005). ACM Press.
[4] Richard C. Atkinson, Edward E. Smith, Daryl J. Bem, and Susan Nolen-Koeksema, *Hilgard's Introduction to Psychology*, Harcourt Brace College Publishers, 12'th edition, 1996.
[5] Amy R. Greenwald and Jeffrey O. Kephart, 'Shopbots and pricebots', in *Agent Mediated Electronic Commerce (IJCAI Workshop)*, pp. 1–23, (1999).

[6] Jeffrey O. Kephart, James E. Hanson, and Jakka Sairamesh, 'Price-war dynamics in a free-market economy of software agents', in *ALIFE: Proceedings of the sixth international conference on Artificial life*, pp. 53–62, Cambridge, MA, USA, (1998). MIT Press.
[7] Long-Ji Lin, 'Programming robots using reinforcement learning and teaching.', in *AAAI*, pp. 781–786, (1991).
[8] Long-Ji Lin, 'Self-improving reactive agents based on reinforcement learning, planning and teaching', *Machine Learning*, **8**(3-4), 293–321, (1992).
[9] Long-Ji Lin, *Reinforcement Learning for Robots using Neural Networks*, Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, 1993. CMU-CS-93-103.
[10] Maja J. Mataric, 'Reward functions for accelerated learning', in *International Conference on Machine Learning*, pp. 181–189, (1994).
[11] Tom Mitchell, *Machine Learning*, McGraw-Hill, 1997.
[12] Jette Randlov and Preben Alstrom, 'Learning to drive a bicycle using reinforcement learning and shaping', in *Machine Learning: Proceedings of the Fifteenth International Conference (ICML'98)*, p. 117. MIT Press, (1998).
[13] Burrhus F. Skinner, *The Behavior of Organisms: An Experimental Analysis*, Prentice Hall, Englewood Cliffs, New Jersey, 1938.
[14] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, 1998.
[15] Ming Tan, 'Multi-agent reinforcement learning: Independent vs. cooperative learning', in *Readings in Agents*, eds., Michael N. Huhns and Munindar P. Singh, 487–494, Morgan Kaufmann, San Francisco, CA, USA, (1997).
[16] Gerald Tesauro, 'Temporal difference learning and td-gammon', *Commun. ACM*, **38**(3), 58–68, (1995).
[17] Gerald Tesauro, 'Pricing in agent economies using neural networks and multi-agent Q-learning', *Lecture Notes in Computer Science*, **1828**, 288–400, (2001).
[18] Steven D. Whitehead, 'A complexity analysis of cooperative mechanisms in reinforcement learning.', in *AAAI*, pp. 607–613, (1991).
[19] Steven D. Whitehead, *Reinforcement Learning for the Adaptive Control of Perception and Action*, Ph.D. dissertation, University of Rochester, 1992.

Shared Intentional Plans for Imitation and Cooperation: Integrating Clues from Child Development and Neurophysiology into Robotics

Peter Ford Dominey

Abstract. One of the long-term goals in the domain of human-robot interaction is that robots will approach these interactions equipped with some of the same fundamental cognitive capabilities that humans use. This will include the ability to perceive and understand human action in terms of an ultimate goal, and more generally to represent shared intentional plans in which the goal directed actions of the robot and the human are interlaced into a shared representation of how to achieve a common goal in a cooperative manner. The current research takes specific experimental protocols from studies of cognitive development to define behavior milestones for a perceptual-motor robotic system. Based on a set of previously established principals for defining the "innate" functions available to such a system, a cognitive architecture is developed that allows the robot to perform cooperative tasks at the level comparable to that of an 18 month old human child. Structural and functional properties of the primate neurophysiological mechanisms for action processing are used to provide further constraints on how the architecture is implemented. At the interface of cognitive development and robotics, the results on cooperation and imitation provide (1) a concrete demonstration of how cognitive neuroscience and developmental studies can contribute to human-robot interaction fidelity, and (2) a demonstration of how robots can be used to experiment with theories on the implementation of cognition in the developing human.

1. INTRODUCTION

One of the current open challenges in cognitive computational neuroscience is to understand the neural basis of the human ability to observe and imitate action. The results from such an endeavor can then be implemented and tested in robotic systems. Recent results from human and non-human primate behavior, neuroanatomy and neurophysiology provide a rich set of observations that allow us to constrain the problem of how imitation is achieved. The current research identifies and exploits constraints in these three domains in order to develop a system for goal directed action perception and imitation.

An impressive body of research exists on human imitation (62K responses to "human imitation" in Google Scholar), which has been empirically studied for over 100 years [15]. One of the recurrent findings across these studies is that in the context of goal directed action, it is the goal itself that tends to take precedence in defining what is to be imitated, rather than the means [1, 6, 25, 28, 29]. Of course in some situations it is the details (e.g. kinematics) of the movement itself that are to be imitated (see discussion in [6, 7]), but the current research focuses on goal based imitation. This body of research helped to formulate questions concerning what could be the neurophysiological substrates for goal based imitation. In 1992 di Pellegrino in the Rizzolatti lab [8] published the first results on "mirror" neurons, whose action potentials reflected both the production of specific goal-directed action, and the perception of the same action being carried by the experimenter. Since then, the premotor and parietal mirror system has been studied in detail in monkey (by single unit recording) and in man (by PET and fMRI) [see 25 for review].

In the context of understanding imitation, the discovery of the mirror system had an immense theoretical impact, as it provided justification for a common code for action production and perception. In recent years a significant research activity has used simulation and robotic platforms to attempt to link imitation behavior to the underlying neurophysiology at different levels of detail (see [24 and 27] for recent reviews from different perspectives, edited volumes [22, 23], and a dedicated special issue of Neural Networks [2]). Such research must directly address the question of how to determine what to imitate. Carpenter and Call [6] distinguish three aspects of the demonstration to copy: the physical action, the resulting change in physical state, and the inferred goal – the internal representation of the desired state. Here we concentrate on imitation of the goal, with the advantage of eliminating the difficulties of mapping detailed movement trajectories across the actor and imitator [7].

Part of the novelty of the current research is that it will explore imitation in the context of cooperative activity in which two agents act in a form of turn-taking sequence, with the actions of each one folding into an interleaved and coordinated intentional action plan. We use the term "shared intentional plan" to insist on the idea that multiple agents have a shared intention that will be realized through their use of a corresponding plan – the shared intentional plan. With respect to constraints derived from behavioral studies, we choose to examine child development studies, because such studies provide well-specified protocols that test behavior that is both relatively simple, and pertinent. The expectation is that a system that can account for this behavior should extend readily to more complex behavior, as demonstrated below.

Looking to the developmental data, Warneken, Chen and Tomasello [31] engaged 18-24 month children and young chimpanzees in goal-oriented tasks and social games which required cooperation. They were interested both in how the cooperation would proceed under optimal conditions, but also how the children and chimps would respond when the adult had a problem in performing the task. The principal finding was that children enthusiastically participate both in goal directed

P. F. Dominey is with the CNRS, 67 Bd Pinel 69675 Bron Cedex, France (phone: 33-437-911266; fax: 33-437-9112110; e-mail: dominey@isc.cnrs.fr).

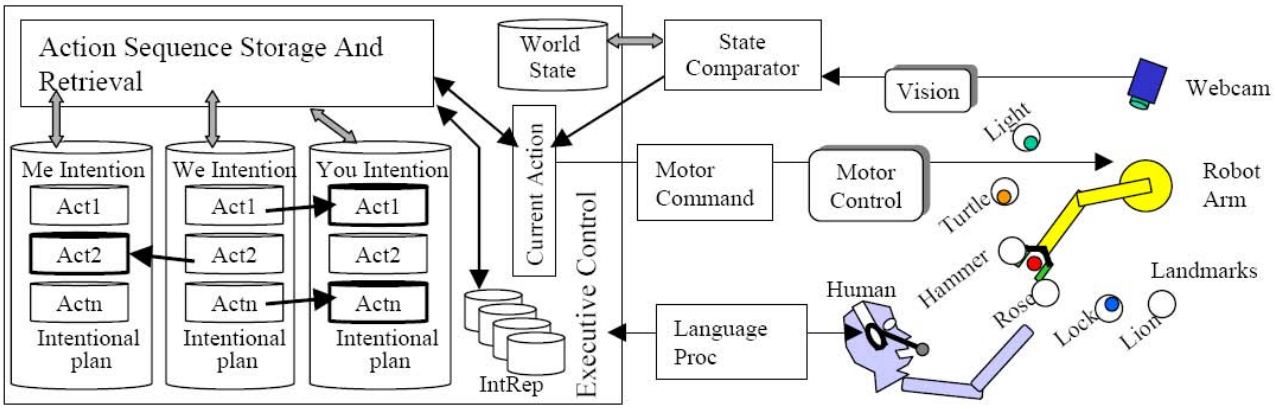


Fig 1. Cooperation System. In a shared work-space, human and robot manipulate objects (green, yellow, read and blue circles corresponding to dog, horse, pig and duck), placing them next to the fixed landmarks (light, turtle, hammer, etc.). *Action*: Spoken commands interpreted as individual words or grammatical constructions, and the command and possible arguments are extracted using grammatical constructions in Language Proc. The resulting Action(Agent, Object, Recipient) representation is the Current Action. This is converted into robot command primitives (Motor Command) and joint angles (Motor Control) for the robot. *Perception*: Vision provides object location input, allowing action to be perceived as changes in World State (State Comparator). Resulting Current Action used for action description, imitation, and cooperative action sequences. *Imitation*: The user performed action is perceived and encoded in Current Action, which is then used to control the robot under the supervision of Executive Control. *Cooperative Games*. During observations, individual actions are perceived, and attributed to the agent or the other player (Me or You). The action sequence is stored in the We Intention structure, that can then be used to separately represent self vs. other actions..

cooperative tasks and social games, and spontaneously attempt to reengage and help the adult when he falters. In contrast, chimps are uninterested in non-goal directed social games, and appear wholly fixed on attaining food goals, independent of cooperation. Warneken et al. thus observed what appears to be a very early human capacity for (1) actively engaging in cooperative activities for the sake of cooperation, and (2) for helping or reengaging the perturbed adult [30, 31].

In one of the social games, the experiment began with a demonstration where one participant sent a wooden block sliding down an inclined tube and the other participant caught the block in a tin cup that made a rattling sound. This can be considered more generally as a task in which one participant moves an object so that the second participant can then in turn manipulate the object. This represents a minimal case of a coordinated action sequence. After the demonstration, in Trials 1 and 2 the experimenter sent the block down one of the tubes three times, and then switched to the other, and the child was required to choose the same tube as the partner. In Trials 3 and 4 during the game, the experimenter interrupted the behavior for 15 seconds and then resumed.

Behaviorally, children successfully participated in the game in Trials 1 and 2. In the interruption Trials 3 and 4 they displayed two particularly interesting types of response that were (a) to attempt to perform the role of the experimenter themselves, and/or (b) to reengage the experimenter with a communicative act. This indicates that the children had a clear awareness both of their role and that of the adult in the shared coordinated activity. This research thus identifies a set of behavioral objectives for robot behavior in the perception and execution of cooperative intentional action. Such behavior could, however, be achieved in a number of possible architectures.

In order to begin to constrain the space of possible solutions we can look to recent results in human and primate neurophysiology and neuroanatomy. It has now become clearly established that neurons in the parietal cortex and the premotor cortex encode the goal of simple actions both for the execution of these actions as well as for the perception of these same goal-directed actions when performed by a second agent [8, 25]. This research thus corroborates the emphasis from behavioral studies on the importance of the goal (rather than the details of the means) in action perception [1, 6, 25, 28, 29]. It has been suggested that these "mirror" neurons play a crucial role in imitation, as they provide a common representation for the perception and subsequent execution of a given action. Interestingly, however, it has been clearly demonstrated that the imitation ability of non-human primates is severely impoverished when compared to that of humans [25, 29-31]. This indicates that the human ability to imitate novel actions and action sequences in real time (i.e. after only one or two demonstrations) relies on additional neural mechanisms.

In this context, a recent study of human imitation learning [5] implicates Brodmann's area (BA) 46 as responsible for orchestrating and selecting the appropriate actions in novel imitation tasks. We have recently proposed that BA 46 participates in a dorsal stream mechanism for the manipulation of variables in abstract sequences and language [14]. Thus, variable "slots" that can be instantiated by arbitrary motor primitives during the observation of new behavior sequences, are controlled in BA 46, and their sequential structure is under the control of corticostriatal systems which have been clearly implicated in sensorimotor sequencing (see [14]). This allows us to propose that this evolutionarily more recent cortical area BA 46 may play a crucial role in allowing humans to perform compositional operations (i.e. sequence learning) on more primitive action representations in the ventral premotor and parietal motor cortices. In other words, ventral premotor and parietal cortices instantiate shared perceptual and motor representations of atomic actions, and BA46 provides the capability to compose arbitrary sequences of these atomic actions, while relying on well known corticostriatal neurophysiology for sequence storage and retrieval. The functional result is the human ability to observe and represent novel behavioral action sequences. We further claim that this system can represent behavioral sequences from the "bird's eye view" or third person perspective, as required for the cooperative tasks of Warneken et al. [31]. That is, it can allow one observer to perceive and form an integrated representation of the coordinated

actions of two other agents engaged in a cooperative activity. The observer can then use this representation to step in and play the role of either of the two agents.

2. IMPLEMENTATION

In a comment on Tomasello et al [29] on understanding and sharing intention, Dominey [10] analyses how a set of initial capabilities can be used to provide the basis for shared intentions. This includes capabilities to

1. perceive the physical states of objects,
2. perceive (and perform) actions that change these states,
3. distinguish between self and other,
4. perceive emotional/evaluation responses in others, and
5. learn sequences of predicate-argument representations.

The goal is to demonstrate how these 5 properties can be implemented within the constraints of the neurophysiology data reviewed above in order to provide the basis for performing these cooperative tasks. In the current experiments the human and robot cooperate by moving physical objects to different positions in a shared work-space as illustrated in Figures 1 and 2. The 4 moveable objects are pieces of a wooden puzzle, representing a dog, a pig, a duck and a cow. These pieces can be moved by the robot and the user in the context of cooperative activity. Each has fixed to it a vertically protruding metal screw, which provides an easy grasping target both for the robot and for humans. In addition there are 6 images that are fixed to the table and serve as landmarks for placing the moveable objects, and correspond to a light, a turtle, a hammer, a rose, a lock and a lion, as partially illustrated in Figures 1 & 2. In the interactions, human and robot are required to place objects in zones next to the different landmarks, so that the robot can more easily determine where objects are, and where to grasp them. Figure 1 provides an overview of the architecture, and Figure 2, which corresponds to Experiment 6 provides an overview of how the system operates.

2.1 Representation

The structure of the internal representations is a central factor determining how the system will function, and how it will generalize to new conditions. Based on the neurophysiology reviewed above, we use a common representation of action for both perception and production. Actions are identified by the agent, the object, and the target location to move that object to. As illustrated in Figure 1, by taking the short loop from vision, via Current Action Representation, to Motor Command, the system is thus configured for a form of goal-centered action imitation. This will be expanded upon below.

A central feature of the system is the World Model that represents the physical state of the world, and can be accessed and updated by vision, motor control, and language, similar to the Grounded Situation Model of [21]. The World Model encodes the physical locations of objects that is updated by vision and proprioception (i.e. robot action updates World Model with new object location). Changes in the World Model in terms of an object being moved allows the system to detect actions in terms these object movements. Actions are represented in terms of the agent, the object and the goal of the action, in the form MOVE(object, goal location, agent). These representations can be used for commanding action, for describing recognized action, and thus for action imitation and narration, as seen below.

In order to allow for more elaborate cooperative activity, the system must be able to store and retrieve actions in a sequential structure. This form of real time sequence learning for imitation is not observed in non-human primates. Interestingly, in this context, an fMRI study [5] that addressed the human ability to observe and program arbitrary actions indicated that a cortical area (BA46) which is of relatively recent phylogenetic origin is involved in such processes. Rizzolatti and Craighero [25] have thus suggested that the BA 46 in man will orchestrate allow the real-time capability to store and retrieve recognized actions, and we can further propose that this orchestration will recruit canonical brain circuitry for sequence processing including the cortico-striatal system (see [14] for discussion of such sequence processing).

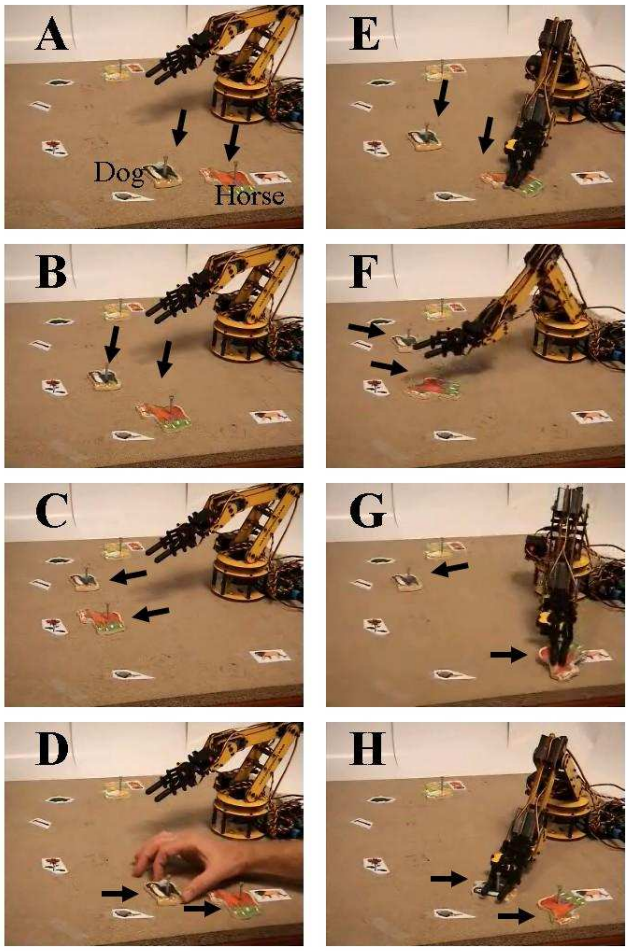


Figure 2. Cooperative task of Exp 5-6. Robot arm, with 6 landmarks (Light, turtle, hammer, rose, lock and lion from top to bottom). Moveable objects include Dog and Horse. In A-D, human demonstrates a "horse chase the dog" game, and successively moves the Dog then Horse, indicating that in the game, the user then the robot are agents, respectively. After demonstration, human and robot "play the game". In each of E – F user moves Dog, and robot follows with Horse. In G robot moves horse, then in H robot detects that the user is having trouble and so "helps" the user with the final move of the dog. See Exp 5 & 6.

In the current study we address behavioral conditions in which focus on the observation and immediate re-use of an intentional (goal directed) action plan. However, in the more general case, one should consider that multiple intentional action plans can be observed and stored in a repertory (IntRep or Intentional Plan Repertory in Figure 1). When the system is subsequently observing the behavior of others, it can compare the ongoing behavior to these stored sequences. Detection of a match

with the beginning of a stored sequence can be used to retrieve the entire sequence. This can then be used to allow the system to "jump into" the scenario, to anticipate the other agent's actions, and/or to help that agent if there is a problem.

2.2 Visual perception

Visual perception is a challenging technical problem. To simplify, standard lighting conditions and a small set (n = 10) of visual object to recognize are employed (4 moveable objects and 6 location landmarks). A VGA webcam is positioned at 1.25 meters above the robot workspace. Vision processing is provided by the Spikenet Vision System (http://www.spikenet-technology.com/). Three recognition models for each object at different orientations (see Fig. 3) were built with an offline model builder. During real-time vision processing, the models are recognized, and their (x, y) location in camera coordinates are provided. Our vision post-processing eliminates spurious detections and returns the reliable (x, y) coordinates of each moveable object in a file. The nearest landmark is then calculated.

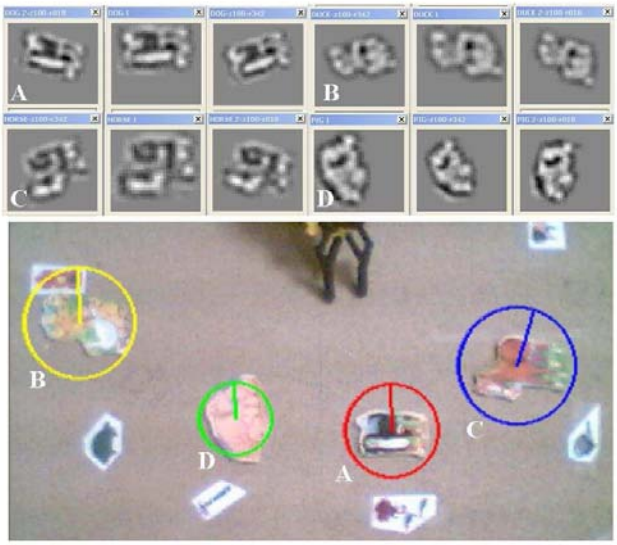


Figure 3. Vision processing. Above: A. – D. Three templates each for the Dog, Duck, Horse and Pig objects at three different orientations. Below, encompassing circles indicate template recognition for the four different objects near different fixed landmarks, as seen from the camera over the robot workspace

2.3 Motor Control & Visual-Motor Coordination

While visual-motor coordination is not the focus of the current work, it was necessary to provide some primitive functions to allow goal directed action. All of the robot actions, whether generated in a context of imitation, spoken command or cooperative interaction will be of the form *move(x to y)* where *x* is a member of a set of visually perceivable objects, and *y* is a member of the set of fixed locations on the work plan.

Robot motor control for transport and object manipulation with a two finger gripper is provided by the 6DOF Lynx6 arm (www.lynxmotion.com). The 6 motors of the arm are coordinated by a parallel controller connected to a PC computer that provides transmission of robot commands over the RS232 serial port.

Human users (and the robot) are constrained when they move an object, to place it in one of the zones designated next to each of the six landmarks (see Fig 3). This way, when the nearest landmark for an object has been determined, this is sufficient for the robot to grasp that object at the prespecified zone.

In a calibration phase, a target point is marked next to each of the 6 fixed landmark locations, such that they are all on an arc that is equidistant to the center of rotation of the robot arm base. For each, the rotation angle of Joint 0 (the rotating shoulder base) necessary to align the arm with that point is then determined, along with a common set of joint angles for Joints 1 – 5 that position the gripper to seize any of the objects. Angles for Joint 6 that controls the closing and opening of the gripper to grasp and release an object were then identified. Finally a neutral position to which the arm could be returned in between movements was defined. The system was thus equipped with a set of primitives that could be combined to position the robot at any of the 6 grasping locations, grasp the corresponding object, move to a new position, and place the object there.

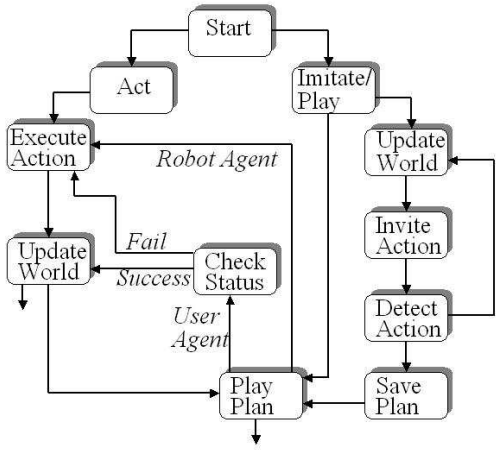


Figure 4. Spoken Language Based Cooperation flow of control. Interaction begins with proposal to act, or imitate/play a game. Act – user says an action that is verified and executed by robot. World Model updated based on action. Downward arrow indicates return to Start. Imitate/Play – user demonstrates actions to robot and says who the agent should be when the game is to be played (e.g. "You/I do this"). Each time, system checks the state of the world, invites the next action and detects the action based on visual object movement. When the demo is finished, the plan (of a single item in the case of imitation) is stored and executed (Play Plan). If the user is the agent (encoded as part of the game sequence), system checks execution status and helps user if failure. If robot is agent, system executes action, and then moves on to next item.

2.4 Cooperation Control Architecture

The spoken language control architecture illustrated in Fig 4 is implemented with the CSLU Rapid Application Development toolkit (http://cslu.cse.ogi.edu/toolkit/). This system provides a state-based dialog management system that allows interaction with the robot (via the serial port controller) and with the vision processing system (via file i/o). It also provides the spoken language interface that allows the user to determine what mode of operation he and the robot will work in, and to manage the interaction via spoken words and sentences.

Figure 4 illustrates the flow of control of the interaction management. In the Start state the system first visually observes

where all of the objects are currently located. From the start state, the system allows the user to specify if he wants to ask the robot to perform actions (Act), to imitate the user, or to play (Imitate/Play). In the Act state, the user can specify actions of the form "Put the dog next to the rose" and a grammatical construction template [9, 11-14] is used to extract the action that the robot then performs.

2.5 Imitation and Learning Shared Intentional Plans

In the Imitate state, the robot first verifies the current state (Update World) and then invites the user to demonstrate an action (Invite Action). The user shows the robot one action. The robot re-observes the world and detects the action based on changes detected (Detect Action). In particular, it will observe that an object has been moved to a new location. This corresponds to the action of moving the object to that location. This action is then saved and transmitted (via Play the Plan with Robot as Agent) to execution (Execute action). A *predicate(argument)* representation of the form Move(object, landmark) is used both for action observation and execution, thus radically simplifying the correspondence problem [see 27]. Imitation is thus a minimal case of Playing in which the "game" is a single action executed by the robot.

In the more general case, the robot should learn to play a game that involves a succession of moves executed by the user and robot in a specific turn-taking sequence. For a given game, the user can demonstrate multiple successive actions, and indicate the agent - by saying "You/I do this" - for each action. The resulting intentional plan specifies what is to be done by whom. When the user specifies that the plan is finished, the system moves to the Save Plan. In this state, the system stores the shared intentional plan, consisting of a sequence of actions and a specification of the agent for each of these action. Control then moves on to the Play Plan state. For each action, the system recalls whether it is to be executed by the robot or the user. Robot execution takes the standard Execute Action pathway. User execution performs a check (based on user response) concerning whether the action was correctly performed or not. If the user action is not performed, then the robot communicates with the user, and performs the action itself. Thus, "helping" was implemented by combining an evaluation of the user action, with the existing capability to perform a stored action representation.

Once the shared intentional plan has been stored or "learned" it can then be re-used in the future. This, when entering the Imitate/Play state, the user is given the option of playing the most recently learned game, or learning a new one.

3. EXPERIMENTAL RESULTS

For each of the 6 following experiments, equivalent variants were repeated at least ten times to demonstrate the generalized capability and robustness of the system. In less than 5 percent of the trials, errors of two types were observed to occur. Speech errors resulted from a failure in the voice recognition, and were recovered from by the command validation check (Robot: "Did you say …?"). Visual image recognition errors occurred when the objects were rotated beyond 20° from their upright position. These errors were identified when the user detected that an object that should be seen was not reported as visible by the system, and were corrected by the user re-placing the object and asking the system to "look again". At the beginning of each trial the system first queries the vision system, and updates the World Model with the position of all visible objects. It then informs the user of the locations of the different objects, for example "The dog is next to the lock, the horse is next to the lion." It then asks the user "Do you want me to act, imitate, play or look again?", and the user responds with one of the action-related options, or with "look again" if the scene is not described correctly.

3.1 Experiment 1: Validation of Sensorimotor Control

In this experiment, the user says that he wants the "Act" state (Fig 4), and then uses spoken commands such as "Put the horse next to the hammer". Recall that the horse is among the moveable objects, and hammer is among the fixed landmarks. The robot requests confirmation and then extracts the predicate-argument representation - *Move(X to Y)* - of the sentence based on grammatical construction templates. In the Execute Action state, the action *Move(X to Y)* is decomposed into two movement primitives [27] of *Get(X)*, and *Place-At(Y)*. *Get(X)* queries the World Model in order to localize X with respect to the different landmarks, and then performs a grasp at the corresponding landmark target location. Likewise, *Place-At(Y)* simply performs a transport to target location Y and releases the object. Decomposing the *get* and *place* functions allows the composition of all possible combinations in the *Move(X to Y)* space. Ten trials were performed moving the four objects to and from different landmark locations. Experiment 1 thus demonstrates (1) the ability to transform a spoken sentence into a Move(X to Y) command, (2) the ability to perform visual localization of the target object, and (3) the sensory-motor ability to grasp the object and put it at the specified location. In ten experimental runs, the system performed correctly.

3.2 Experiment 2: Imitation

In this experiment the user chooses the "imitate" state. As stated above, imitation is centered on the achieved ends – in terms of observed changes in state – rather than the means towards these ends. Before the user performs the demonstration of the action to be imitated, the robot queries the vision system, and updates the World Model (Update World in Fig 4) and then invites the user to demonstrate an action. The robot pauses, and then again queries the vision system and continues to query until it detects a difference between the currently perceived world state and the previously stored World Model (in State Comparator of Fig 1, and Detect Action in Fig 4), corresponding to an object displacement. Extracting the identity of the displaced object, and its new location (with respect to the nearest landmark) allows the formation of an *Move(object, location)* action representation. Before imitating, the robot operates on this representation with a meaning-to-sentence construction in order to verify the action to the user, as in "Did you put the dog next to the rose?" It then asks the user to put things back as they were so that it can perform the imitation. At this point, the action is executed (Execute Action in Fig 4). In ten experimental runs the system performed correctly. This demonstrates (1) the ability of the system to detect the goals of user-generated actions based on visually perceived state changes, and (2) the utility of a common representation of action for perception, description and execution.

3.3 Experiment 3: A Cooperative Game

The cooperative game is similar to imitation, except that there is a sequence of actions (rather than just one), and the actions can be effected by either the user or the robot in a cooperative manner. In this experiment, the user responds to the system request and enters the "play" state. In what corresponds to the demonstration in Warneken et al. [17] the robot invites the user to start showing how the game works. The user then begins to perform a sequence of actions. For each action, the user specifies who does the action, i.e. either "you do this" or "I do this". The intentional plan is thus stored as a sequence of action-agent pairs, where each action is the movement of an object to a particular target location. In Fig 1, the resulting interleaved sequence is stored as the "We intention", i.e. an action sequence in which there are different agents for different actions. When the user is finished he says "play the game". The robot then begins to execute the stored intentional plan. During the execution, the "We intention" is decomposed into the components for the robot (Me Intention) and the human (You intention).

In one run, during the demonstration, the user said "I do this" and moved the horse from the lock location to the rose location. He then said "you do this" and moved the horse back to the lock location. After each move, the robot asks "Another move, or shall we play the game?". When the user is finished demonstrating the game, he replies "Play the game." During the playing of this game, the robot announced "Now user puts the horse by the rose". The user then performed this movement. The robot then asked the user "Is it OK?" to which the user replied "Yes". The robot then announced "Now robot puts the horse by the lock" and performed the action. In two experimental runs of different demonstrations, and 5 runs each of the two demonstrated games, the system performed correctly. This demonstrates that the system can learn a simple intentional plan as a stored action sequence in which the human and the robot are agents in the respective actions.

Action	User identifies agent	User Demonstrates Action	Ref in Figure 2
1.	I do this	Move dog from the lock to the rose	B
2.	You do this	Move the horse from the lion to the lock	B
3.	I do this	Move the dog from the rose to the hammer	C
4.	You do this	Move the horse from the lock to the rose	C
5.	You do this	Move the horse from the rose to the lion	D
6.	I do this	Move the dog from the hammer to the lock	D

Table 1. Cooperative "horse chase the dog" game specified by the user in terms of who does the action (indicated by saying) and what the action is (indicated by demonstration). Illustrated in Figure 2.

3.4 Experiment 4: Interrupting a Cooperative Game

In this experiment, everything proceeds as in experiment 3, except that after one correct repetition of the game, in the next repetition, when the robot announced "Now user puts the horse by the rose" the user did nothing. The robot asked "Is it OK" and during a 15 second delay, the user replied "no". The robot then said "Let me help you" and executed the move of the horse to the rose. Play then continued for the remaining move of the robot. This illustrates how the robot's stored representation of the action that was to be performed by the user allowed the robot to "help" the user.

3.5 Experiment 5: A More Complex Game

Experiment 3 represented the simplest behavior that could qualify as a cooperative action sequence. In order to more explicitly test the intentional sequencing capability of the system, this experiment replicates Exp 3 but with a more complex task, illustrated in Figure 2. In this game (Table 1), the user starts by moving0 the dog, and after each move the robot "chases" the dog with the horse, till they both return to their starting places.

As in Experiment 3, the successive actions are visually recognized and stored in the shared "We Intention" representation. Once the user says "Play the game", the final sequence is stored, and then during the execution, the shared sequence is decomposed into the robot and user components based on the agent associated with each action. When the user is the agent, the system invites the user to make the next move, and verifies (by asking) if the move was ok. When the system is the agent, the robot executes the movement. After each move the World Model is updated. As in Exp 3, two different complex games were learned, and each one "played" 5 times. This illustrates the learning by demonstration [31] of a complex intentional plan in which the human and the robot are agents in a coordinated and cooperative activity.

3.6 Experiment 6: Interrupting the Complex Game

As in Experiment 4, the objective was to verify that the robot would take over if the human had a problem. In the current experiment this capability is verified in a more complex setting. Thus, when the user is making the final movement of the dog back to the "lock" location, he fails to perform correctly, and indicates this to the robot. When the robot detects failure, it reengages the user with spoken language, and then offers to fill in for the user. This is illustrated in Figure 2H. This demonstrates the generalized ability to help that can occur whenever the robot detects the user is in trouble.

4. DISCUSSION

Significant progress has been made in identifying some of the fundamental characteristics of human cognition in the context of cooperative interaction, particularly with respect to social cognition [16-19]. Breazeal and Scassellati [4] investigate how perception of socially relevant face stimuli and object motion will both influence the emotional and attentional state of the system and thus the human-robot interaction. Scassellati [26] further investigates how developmental theories of human social cognition can be implemented in robots. In this context, Kozima and Yano [18] outline how a robot can attain intentionality – the linking of goal states with intentional actions to achieve those goals – based on innate capabilities including: sensory-motor function and a simple behavior repertoire, drives, an evaluation function, and a learning mechanism.

The abilities to observe an action, determine its goal and attribute this to another agent are all clearly important aspects of the human ability to cooperate with others. The current research demonstrates how these capabilities can contribute to the "social" behavior of learning to play a cooperative game, playing the game, and helping another player who has gotten stuck in the game, as

displayed in 18-24 month children [29, 30]. While the primitive bases of such behavior is visible in chimps, its full expression is uniquely human [29, 30]. As such, it can be considered a crucial component of human-like behavior for robots.

The current research is part of an ongoing effort to understand aspects of human social cognition by bridging the gap between cognitive neuroscience, simulation and robotics [3, 9-14], with a focus on the role of language (see [20]). The experiments presented here indicate that functional requirements derived from human child behavior and neurophysiological constraints can be used to define a system that displays some interesting capabilities for cooperative behavior in the context of imitation. Likewise, they indicate that evaluation of another's progress, combined with a representation of his/her failed goal provides the basis for the human characteristic of "helping." This may be of interest to developmental scientists, and the potential collaboration between these two fields of cognitive robotics and human cognitive development is promising. The developmental cognition literature lays out a virtual roadmap for robot cognitive development [10, 28]. In this context, we are currently investigating the development of hierarchical means-end action sequences [27]. At each step, the objective will be to identify the behavior characteristic and to implement it in the most economic manner in this continuously developing system for human-robot cooperation.

At least two natural extensions to the current system can be considered. The first involves the possibility for changes in perspective. In the experiments of Warneken et al. the child watched two adults perform a coordinated task (one adult launching the block down the tube, and the other catching the block). At 24 months, the child can thus observe the two roles being played out, and then step into either role. This indicates a "bird's eye view" representation of the cooperation, in which rather than assigning "me" and "other" agent roles from the outset, the child represents the two distinct agents A and B for each action in the cooperative sequence. Then, once the perspective shift is established (by the adult taking one of the roles, or letting the child choose one) the roles A and B are assigned to me and you (or vice versa) as appropriate.

This actually represents a minimal change to our current system. First, rather than assigning the "you" "me" roles in the We Intention at the outset, these should be assigned as A and B. Then, once the decision is made as to the mapping of A and B onto robot and user, these agent values will then be assigned accordingly. Second, rather than having the user tell the robot "you do this" and "I do this" the vision system can be modified to recognize different agents who can be identified by saying their name as they act, or via visually identified cues on their acting hands.

The second issue has to do with inferring intentions. The current research addresses one cooperative activity at a time, but nothing prevents the system from storing multiple such intentional plans in a repertory (IntRep in Fig 1). In this case, as the user begins to perform a sequence of actions involving himself and the robot, the robot can compare this ongoing sequence to the initial subsequences of all stored sequences in the IntRep. In case of a match, the robot can retrieve the matching sequence, and infer that it is this that the user wants to perform. This can be confirmed with the user and thus provides the basis for a potentially useful form of learning for cooperative activity.

In conclusion, the current research has attempted to build and test a robotic system for interaction with humans, based on behavioral and neurophysiological requirements derived from the respective literatures. The interaction involves spoken language and the performance and observation of actions in the context of cooperative action. The experimental results demonstrate a rich set of capabilities for robot perception and subsequent use of cooperative action plans in the context of human-robot cooperation. This work thus extends the imitation paradigm into that of sequential behavior, in which the learned intentional action sequences are made up of interlaced action sequences performed in cooperative alternation by the human and robot. While many technical aspects of robotics (including visuomotor coordination and vision) have been simplified, it is hoped that the contribution to the study of imitation and cooperative activity is of some value.

5. ACKNOWLEDGEMENTS

I thank Mike Tomasello, Felix Warneken, Malinda Carpenter and Elena Lieven for useful discussions during a visit to the MPI EVA in Leipzig concerning shared intentions; and Giacomo Rizzolatti for insightful discussion concerning the neurophysiology of sequence imitation at the IEEE Humanoids meeting in Genoa 2006. This research is supported in part by the French Minister of Research under grant ACI-TTT, ACI-NIC and by the LAFMI.

6. REFERENCES

[1] Bekkering H, WohlschlagerA , Gattis M (2000) Imitation of Gestures in Children is Goal-directed, The Quarterly Journal of Experimental Psychology: Section A, 53, 153-164

[2] Billard A, Schaal (2006) Special Issue: The Brain Mechanisms of Imitation Learning, Neural Networks, 19(1) 251-338

[3] Boucher J-D, Dominey PF (2006) Programming by Cooperation: Perceptual-Motor Sequence Learning via Human-Robot Interaction, Proc. Simulation of Adaptive Behavior, Rome 2006.

[4] Breazeal C., Scassellati B., (2001) Challenges in building robots that imitate people, in: K. Dautenhahn, C. Nehaniv (Eds.), Imitation in Animals and Artifacts, MIT Press, Cambridge, MA,.

[5] Buchine G, Vogt S, Ritzl A, Fink GR, Zilles K, Freund H-J, Rizzolatti G (2004) Neural circuits Underlying Imitation Learning of Hand Actions: An Event-Related fMRI Study. Neuron, (42) 323-334.

[6] Carpenter M, Call Josep (2007) The question of 'what to imitate': inferring goals and intentions from demonstrations, in Chrystopher L. Nehaniv and Kerstin Dautenhahn Eds, *Imitation and Social Learning in Robots, Human sand Animals*, Cambridge Univerity Press, Cambridge.

[7] Cuijpers RH, van Schie HT, Koppen M, Erlhagen W, Bekkering H (2006) Goals and means in action observation: A computational approach, *Neural Networks* 19, 311-322,

[8] di Pellegrino G, Fadiga L, Fogassi L, Gallese V, Rizzolatti G (1992) Understanding motor events: a neurophysiological study. *Exp Brain Res*.;91(1):176-80.

[9] Dominey, P.F., (2003) Learning grammatical constructions from narrated video events for human–robot interaction. *Proceedings IEEE Humanoid Robotics Conference*, Karlsruhe, Germany

[10] Dominey PF (2005) Toward a construction-based account of shared intentions in social cognition. Comment on Tomasello et al. 2005, *Beh Brain Sci*. 28:5, p. 696.

[11] Dominey PF, Alvarez M, Gao B, Jeambrun M, Weitzenfeld A, Medrano A (2005) Robot Command, Interrogation and Teaching via Social Interaction, *Proc. IEEE Conf. On Humanoid Robotics 2005*.

[12] Dominey PF, Boucher (2005) Learning To Talk About Events From Narrated Video in the Construction Grammar Framework, *Artificial Intelligence*, 167 (2005) 31–61

[13] Dominey, P. F., Boucher, J. D., & Inui, T. (2004). Building an adaptive spoken language interface for perceptually grounded human–robot interaction. In *Proceedings of the IEEE-RAS/RSJ international conference on humanoid robots*.

[14] Dominey PF, Hoen M, Inui T. (2006) A neurolinguistic model of grammatical construction processing. *Journal of Cognitive Neuroscience*.18(12):2088-107.

[15] Ellwood CA (1901) The Theory of Imitation in Social Psychology *The American Journal of Sociology*, Vol. 6, No. 6 (May, 1901), pp. 721-741

[16] Fong T, Nourbakhsh I, Dautenhaln K (2003) A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42 3-4, 143-166.

[17] Goga, I., Billard, A. (2005), Development of goal-directed imitation, object manipulation and language in humans and robots. In M. A. Arbib (ed.), *Action to Language via the Mirror Neuron System*, Cambridge University Press (in press).

[18] Kozima H., Yano H. (2001) A robot that learns to communicate with human caregivers, in: *Proceedings of the International Workshop on Epigenetic Robotics*,.

[19] Lieberman MD (2007) Social Cognitive neuroscience: A Review of Core Processes, *Annu. Rev. Psychol*. (58) 18.1-18.31

[20] Lauria S, Buggmann G, Kyriacou T, Klein E (2002) Mobile robot programming using natural language. *Robotics and Autonomous Systems* 38(3-4) 171-181

[21] Mavridis N, Roy D (2006). Grounded Situation Models for Robots: Where Words and Percepts Meet. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*

[22] Nehaniv CL, Dautenhahn K eds. (2002) Imitation in Animals and Artifacts; MIT Press, Cambridge MA.

[23] Nehaniv CL, Dautenhahn K eds. (2007) *Imitation and Social Learing in Robots, Humans and Animals*, Cambridge University Press, Cambridge.

[24] Oztop E, Kawato M, Arbib M (2006) Mirror neurons and imitation: A computationally guided review. *Neural Networks*, (19) 254-271

[25] Rizzolatti G, Craighero L (2004) The Mirror-Neuron system, *Annu. Rev. Neuroscience* (27) 169-192

[26] Scassellati B (2002) Theory of mind for a humanoid robot, *Autonomous Robots*, 12(1) 13-24

[27] Schaal S, Ijspeert A, Billard A (2003) Computational Approaches to Motor Learning by Imitation; Phil Trans Royal Society London/ B, 358: 537-547.

[28] Sommerville A, Woodward AL (2005) Pulling out the intentional structure of action: the relation between action processing and action production in infancy. *Cognition*, 95, 1-30.

[29] Tomasello M, Carpenter M, Cal J, Behne T, Moll HY (2005) Understanding and sharing intentions: The origins of cultural cognition, *Beh. Brain Sc*;. 28; 675-735.

[30] Warneken F, Tomasello M (2006) Altruistic helping in human infants and young chimpanzees, *Science*, 311, 1301-1303

[31] Warneken F, Chen F, Tomasello M (2006) Cooperative Activities in Young Children and Chimpanzees, *Child Development*, 77(3) 640-663.

[32] Zöllner R., Asfour T., Dillman R.: Programming by Demonstration: Dual-Arm Manipulation Tasks for Humanoid Robots. *Proc IEEE/RSJ Intern. Conf on Intelligent Robots and systems (IROS 2004)*.

Multiagent Collaborative Task Learning through Imitation

Sonia Chernova and **Manuela Veloso** [1]

Abstract. Learning through imitation is a powerful approach for acquiring new behaviors. Imitation-based methods have been successfully applied to a wide range of single agent problems, consistently demonstrating faster learning rates compared to exploration-based approaches such as reinforcement learning. The potential for rapid behavior acquisition from human demonstration makes imitation a promising approach for learning in multiagent systems. In this work, we present results from our single agent demonstration-based learning algorithm, aimed at reducing demonstration demand of a single agent on the teacher over time. We then demonstrate how this approach can be applied to effectively train a complex multiagent task requiring explicit coordination between agents. We believe that this is the first application of demonstration-based learning to simultaneously training distinct policies to multiple agents. We validate our approach with experiments in two complex simulated domains.

1 Introduction

Programming robots is a challenging problem due to sensor complexity, noise, and the non-deterministic effects of robot actions. To address this challenge, autonomous learning approaches have been developed that allow robots to learn task execution through interaction with the environment [14]. Most of these approaches, however, rely on a long trial-and-error experimental process that is impractical due to time constraints and physical wear on the robot. Learning in systems with multiple robots is further complicated by the complex interactions that can occur between distributed agents, such as communication via message passing, physical interaction and resource contention. To address these problems, natural and intuitive approaches must be developed that allow new skills to be taught to multiple of robots in a timely manner.

Learning from demonstration, a collaborative learning approach based on human-robot interaction, offers an alternative to exploration-based methods. The goal of this approach is to learn to imitate the behavior of a teacher by watching a demonstration of the task. Demonstration-based learning has been successfully applied to a variety of single agent learning problems [5, 8, 18, 28]; its fast learning rate compared to exploration-based learning methods, such as reinforcement learning, makes learning from demonstration a promising approach for multiagent systems.

In this work, we first present results of our single agent demonstration-based learning algorithm, the *confident execution* framework [10]. We then apply this framework to a collaborative multiagent domain, demonstrating its effectiveness in simultaneously training multiple robots to perform a joint task. Our learning framework aims to reduce each agent's demonstration demands on the teacher by allowing the agent to perform its task autonomously when it is confident about its actions, and request expert assistance at times of uncertainty. As a result, each agent operates with gradually increasing autonomy as the task is learned, relieving the teacher from repeated demonstrations of acquired behavior and allowing simultaneous supervision of multiple agents.

In the next section we discuss related work in the areas of demonstration and imitation learning, followed by a complete description of the confident execution learning framework in Section 3. In Section 4 we present experimental results demonstrating our approach in single and multi agent domains.

2 Related Work

Learning from demonstration is an interactive learning method in which the agent aims to imitate the behavior of an expert teacher. Demonstration-based methods have been successfully applied to a wide range of single agent learning problems.

Nicolescu and Mataric [17, 18] present a learning framework based on demonstration, generalization and teacher feedback, in which training is performed by having the robot follow a human and observe its actions. A high-level task representation is then constructed by analyzing the experience with respect to the robot's underlying capabilities. The authors also describe a generalization of the framework that allows the robot to interactively request help from a human in order to resolve problems and unexpected situations. This interaction is implicit as the agent has no direct method of communication; instead, it attempts to convey its intentions by communicating though its actions.

Lockerd and Breazeal [8, 15] demonstrate a robotic system where high-level tasks are taught through social interaction. In this framework, the teacher interacts with the agent through speech and visual inputs, and the learning agent expresses its internal state through emotive cues such as facial and body expressions to help guide the teaching process. The outcome of the learning is a goal-oriented hierarchical task model.

Bentivegna et al. [5, 6, 7] and Saunders et al. [25] present demonstration learning approaches using memory-based techniques. Both groups use the k-nearest neighbor (KNN) [16] algorithm to classify instances based on similarity to training examples, resulting in a policy mapping from sensory observations to actions. Our algorithm takes a similar approach by utilizing Gaussian mixture models for classification, but includes an interactive learning component similar to Nicolescu and Mataric. Inamura et al. [13] present a similar method based on Bayesian Networks [20] limited to a discretely-

[1] Computer Science Department, Carnegie Mellon University, email: soniac@cs.cmu.edu, veloso@cs.cmu.edu

valued feature set.

A handful of studies have also examined imitation in the context of multiagent systems. In the Ask For Help framework [11], reinforcement learning agents request advice from other similar agents in the environment. Help is requested when an agent is confused about what action to take, an event characterized by relatively equal quality estimates for all possible actions in a given state.

A similar approach is presented by Oliveira and Nunes [19], in which agents are able to select, exchange and incorporate advice from other agents, combining it with reinforcement learning to improve learning performance. The authors examine when and how agents should exchange advice, and which of an agent's teammates should be communicated with. Their results show that exchange of information can improve the average performance of learning agents, although it may reduce the exploration of the state space, preventing the optimal policy from being found in some cases.

Alissandrakis et al. [2, 3] present a general framework that enables a robotic agent to imitate another, possibly differently embodied, agent through observation. Using this framework, the authors demonstrate the transmission of skills between individuals in a heterogeneous community of software agents. Their results indicate that transmission of a behavior pattern through a chain of agents can be achieved despite differences in the embodiment of some agents in the chain. Additionally, the authors show that groups of mutually imitating agents are able to converge to a common shared behavior.

Price and Boutilier [21] present a multiagent system in which novice agents learn by passively observing other agents in the environment. Each learning agent is limited to observing the actions of others and no explicit teaching occurs. By observing a mentor, the reinforcement learning agent can extract information about its own capabilities in, and the relative value of, unvisited parts of the state space. However, the task of an observed agent may be so different that the observations provide little useful information for the learner, in which case direct imitation of this expert must be avoided by the algorithm.

The above methods study imitation from the perspective of a community of agents, where a single agent seeks advice from other members of its group. A different approach is taken in the study of coaching [23], where an external coach agent provides advice to a team of agents in order to improve their performance at a task. The coach has an external, often broader, view of the world and is able to provide advice to the agents, but not control them. The agents must decide how to incorporate the coach's advice into their execution. Riley [23] presents an approach for training the coach using imitation based on example executions.

Our approach differs from the presented techniques in that it enables a single human to simultaneously train multiple agents. The agents may be differently embodied, and may learn different policies and perform different tasks. In our proposed system, the human teacher is the only source of advice, providing demonstrations in the form of action commands.

3 The Confident Execution Framework

In this section, we present a summary of our confident execution learning framework which allows a single agent to learn a task policy from demonstration (for a more detailed description, please see [10]). We then describe how this framework can be applied to simultaneously training multiple robots to perform a joint task.

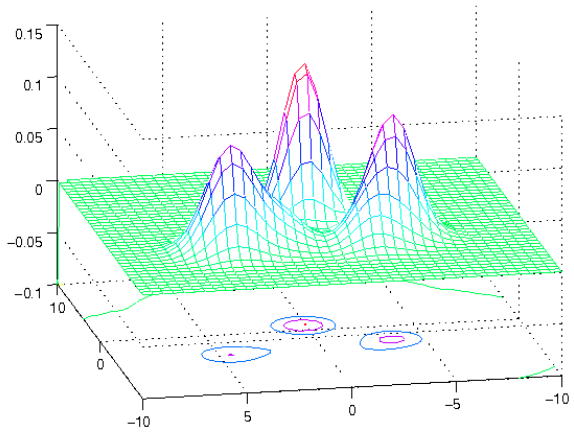


Figure 1. An example of a 2-dimensional Gaussian mixture model with three components. Contour lines below the GMM mark the one- and two-standard deviation ellipses.

3.1 Task Representation

Our approach utilizes the *learning by experienced demonstration* technique [18], in which the agent is fully under the expert's control while continuing to experience the task through its own sensors. During each training timestep, the agent records sensory observations about its environment and executes the action selected by the human expert. We assume the expert attempts to perform the task optimally, without necessarily succeeding.

Observations o are represented using an n-dimensional feature vector that can be composed of continuous or discrete values representing the state of the robot. The agent's actions a are bound to a finite set $\mathcal{A}$ of action primitives [4], which are the basic actions that can be combined together to perform the overall task. The goal of the system is to learn a policy $\pi : o \rightarrow \mathcal{A}$, mapping observations to action primitives. Each labeled training point (o, a) consists of an observation labeled by its corresponding expert-selected action.

During training, the algorithm separates all datapoints into classes based on their action label. A Gaussian mixture model (GMM), Figure 1, is then trained for each action class using the expectation-maximization (EM) algorithm [12]. We selected Gaussian mixture models for our approach due to previous successes of classification methods in demonstration learning [5, 25], and because GMMs provide a built-in measure of classification confidence. Their robustness to noise and ability to generalize and capture correlations between continuous features make GMMs a powerful tool for robotic data analysis.

Since a single action is often associated with a number of distinct states (the action *turn left* may be taken from several different locations), we use a separate Gaussian mixture to represent each action class. Components within the mixture represent the different state regions and the number of components is determined using cross-validation. New datapoints are classified by selecting the Gaussian mixture with the maximum likelihood. The output of the classification is the action represented by the selected GMM. Additionally, the model returns a confidence value representing the certainty of the classification based on the likelihood.

3.2 The Learning Process

Table 1 shows a pseudocode summary of the learning process. Learning begins with a non-interactive demonstration training phase during which each action of the robot is controlled by the expert through teleoperation. The algorithm uses training examples acquired from the demonstrations to generate a task model. Every time $maxNew$ additional training points are acquired, the algorithm updates the GMM based on the new data.

Additionally, the performance of the current learned policy is evaluated by comparing how closely it matches the behavior of the expert. Prior to updating the model with each new training point (o, a), the algorithm classifies observation o using the current model. It then compares the model-selected action to the demonstrated action a. Performing this comparison over a window of consecutive training points results in an estimate of the prediction accuracy of the model that relates how closely the policy matches the behavior of the expert.

The teacher performs non-interactive training until the model prediction accuracy is sufficiently high, as determined by the expert. At this point, learning transitions to the *confident execution* stage, during which the agent selects between autonomously executing its learned policy action and requesting help from the expert based on the classification confidence of the above model. The algorithm adjust the agent's autonomy by comparing the classification confidence to an autonomy threshold. Classification confidences greater than the threshold result in autonomous execution of the model-selected action, while confidences below the threshold cause the agent to pause its execution of the task and signal the teacher that a demonstration is needed.

Since the autonomy threshold value is continuous, our approach allows smooth adjustment of the autonomy level. This type of mechanism is referred to as adjustable, or sliding, autonomy and has been proven effective in a wide range of applications, from personal assistants [26] to space exploration [27]. Our algorithm combines learning with adjustable autonomy, resulting in an interactive teaching method that targets low confidence regions of the state space and reduces dependence on the human expert as the agent gains proficiency at its task. In the presented experiments, the human teacher manually sets the confidence threshold value that determines the level of autonomy. We are currently developing a technique for calculating this value automatically.

As the agent's model improves over time, the agent will encounter fewer observations with low classification confidence, resulting in fewer demonstration requests. Learning terminates when the agent is able to execute the task completely autonomously, or when the expert is satisfied with the performance of the model. The agent then deterministically executes the action selected by the model, regardless of the classification confidence. This mode of operation is typical of traditional learning approaches where the learned policy is always trusted once learning is complete.

3.3 Multiagent Approach

The confident execution learning framework is a promising approach for multiagent learning due to its fast learning rate compared to exploration-based methods such as reinforcement learning [10], and reduced demand on the expert over time. In this work, we examine how it can be directly applied to training multiple agents simultaneously.

In a multiagent setting, the expert's workload and teaching style differ depending on the degree of collaboration required between the agents. Domains with little collaboration allow each agent to operate with little regard for the actions of others, and training can be done independently for each agent. In such cases, it may be possible to introduce new agents at different times, resulting in a mixture of novice and expert agents to avoid overloading the expert at the beginning of the training stage. Domains that require greater collaboration between agents benefit from demonstration-based approaches because exploration over the joint action space of multiple robots is quite costly [9]. In these domains, it is beneficial to demonstrate the task to multiple collaborating agents at the same time.

Using our approach described in the previous section, each agent is able to learn its own individual policy regardless of the level of collaboration required. Our algorithm scales to an arbitrary number of robots without any modifications to the learning framework.

Algorithm 3.1: THE LEARNING FRAMEWORK()

procedure INITIALTRAINING()
 $observation \leftarrow$ GETSENSORDATA()
 $expertAct \leftarrow$ GETEXPERTACTION()
 $(gmmAct, conf) \leftarrow$ CLASSIFY$(observation)$
 $predAccuracy \leftarrow$ TRACKPRED$(gmmAct, expertAct)$

 if $numNewDatapoints >$ maxNew :
  then UPDATEMODEL$(observation, expertAct)$

 EXECUTEACTION$(expertAct)$
 return $(predAccuracy)$

procedure CONFIDENTEXECUTION()
 $observation \leftarrow$ GETSENSORDATA()
 $(gmmAct, conf) \leftarrow$ CLASSIFY$(observation)$

 if $conf > confThresh$:
  then { EXECUTEACTION$(gmmAct)$
  else { $expertAct \leftarrow$ GETEXPERTACTION()
  UPDATEMODEL$(observation, expertAct)$
  EXECUTEACTION$(expertAct)$

Table 1. Pseudocode overview of the learning framework.

4 Experimental Results

We validate our approach using two simulated domains with continuous and multidimensional feature spaces.

4.1 Single Agent Driving Domain

In this section we present results of a fast and dynamic simulated car driving domain (Figure 2). In this domain, the agent takes the shape of a car that must be driven by the expert on a busy road. The speed of the car is fixed at 60 mph while all other cars move in their lanes at predetermined speeds between 20 and 40 mph. The learning agent can not change its speed, and must navigate between other cars to avoid collision. The agent is limited to three actions: remaining in the current lane, and shifting one lane to the left or right of the current position. The road has three normal lanes and a shoulder lane on both sides; the car is allowed to drive on the shoulder but can not go further off the road.

The environment is represented using four features, a distance to the nearest car in each of the three lanes and the current lane of the agent. The agent's lane is represented using a discrete value symbolizing the lane number. The distance features are continuously valued

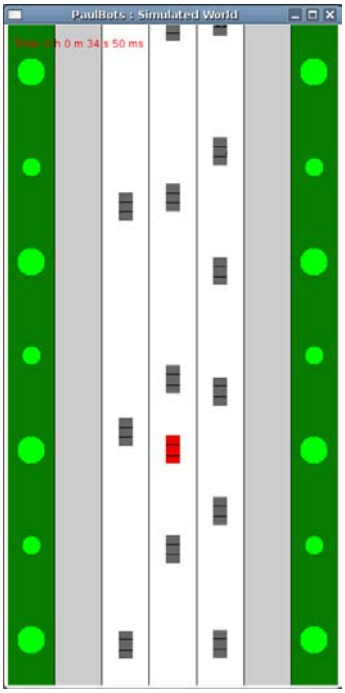


Figure 2. Screenshot of the driving simulator.

in the [-25,25] range; note that the nearest car in a lane can be behind the agent.

Demonstration of the task was performed by a human using a keyboard interface. Figure 3 shows the prediction accuracy of the model during the initial non-interactive training phase. Training was performed until the model reached 80% prediction accuracy over a 150-timestep window, which resulted in a demonstration length of 500 timesteps, or approximately 2.1 minutes. After transitioning to the confident execution phase, the expert completed the training after 150 demonstration timesteps when the model exhibited good performance. During the confident execution phase all demonstrations were done as sequences of ten consecutive moves to simplify the task of the expert due to the fast-paced nature of this domain.

The feature space of this domain is complex as the different action classes frequently overlap. Figure 4 shows a small sample of the data representing how the agent should drive in the middle lane. The data is split into two regions based on the relative position (in front or behind) of the nearest car in the agent's current lane. No samples appear in the 10 to -10 distance range along the Lane2 axis as the expert avoids collisions that would occur from having another car in such close proximity.

The final model consisted of 34 Gaussian components across three GMMs (one for each action class). The final policy was able to imitate the expert's driving style and navigate well in the complex driving domain. Since the algorithm aims to imitate the behavior of the expert, no 'true' reward function exists to evaluate the performance of a given policy. However, we present two domain-specific evaluation metrics that capture the key characteristics of the driving task.

Since the demonstrated behavior attempts to navigate the domain without collisions, our first evaluation metric is the number of collisions experienced under each policy. Collisions are measured as the percentage of the total timesteps that the agent spends in contact with another car. Always driving straight and colliding with every car in the middle lane results in a 30% collision rate.

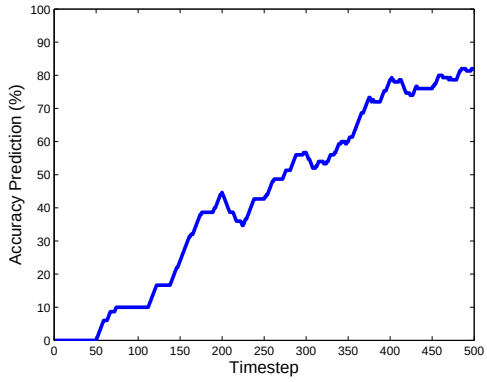


Figure 3. Prediction accuracy of the learned model over the non-interactive training phase using a window of 150 timesteps.

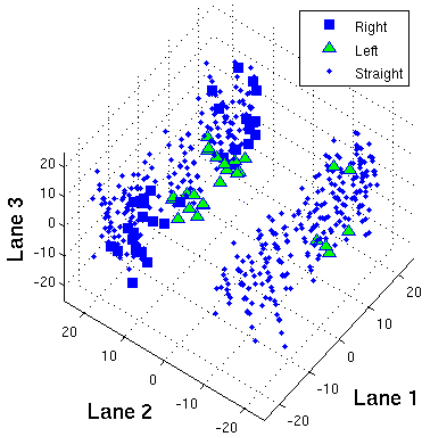


Figure 4. Driving training data representing the driving strategy used when the agent is in the middle lane. Graph axes represent distance to the nearest car in each of the three driving lanes.

Our second evaluation metric is the proportion of the time the agent spends in each lane over the course of a trial. This metric captures the driving preferences of the expert and provides an estimate of the similarity in driving styles. Each evaluation trial was performed for 1000 timesteps over an identical road segment.

Figure 5 compares the performance of the algorithm at different stages in the learning process using these two metrics. Each line in the figure represents a composite bar graph showing the percentage of total time spent by the agent in each lane. Collision percentages for each policy are reported to the right of the bar graphs. The bottom line in the figure shows the performance of the expert over the evaluation road segment (not used for training). We see that the expert successfully avoids collisions, and prefers to use the left three lanes, only rarely using the right lane and right shoulder.

The top five lines summarize the behavior of the agent during the non-interactive training phase. Training was stopped after every 100 training examples for evaluation. Initially the agent always remains in the center lane, accumulating a 30.4% collision rate in the process. As learning progresses, the agent learns to change lanes effectively, beginning to use all five available lanes after 500 demonstration instances, with a collision rate of only 1.3%. However, the agent's lane preference differs significantly from the expert as the agent spends most of its time driving on the right shoulder.

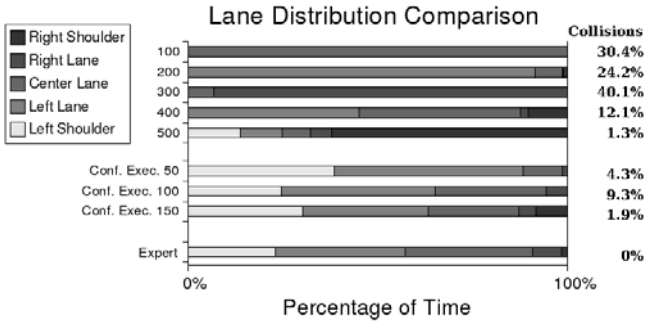


Figure 5. Policy performance comparison using lane distribution and collision evaluation metrics.

The three middle lines display performance during the confident execution phase at 50-timestep intervals. Similarity in lane preference improves over this final training phase, reaching final performance very similar to that of the expert. Additionally, our agent's performance is comparable to that learned using Inverse Reinforcement Learning by Abbeel et al. in [1]. For further evaluation of this domain, including empirical results demonstrating how adapting execution based on confidence focuses training on relevant areas of the domain and a comparison between confident execution and non-interactive demonstration, please see our previous work [10].

4.2 Multiagent Furniture Movers Domain

In this section we present a multiagent collaborative furniture movers domain, Figure 6. In this domain, two agents must move a long, heavy couch from one room to another through a narrow hallway and stairs. We assume that the agents hold opposite ends of the furniture piece throughout this task. Each agent uses six noisy local sensors to determine distances to nearby walls. Additionally, each agent is equipped with a stair sensor that reports a binary value representing the presence or absence of a staircase in the immediate vicinity. The complete feature vector for each agent consists of six continuous distance measurements, and two binary stair features, one for the agent's own location and one for its teammate's. Note that each agent only has a local view of the world, and its teammate's stair information is only updated via a special *communicate* action.

A total of six actions are available to the agents: *forward*, *back*, *left*, *right*, *communicate*, and *stair*. At each timestep, each agent executes an action based on its own individual policy, and their overall movement is determined by the joint action of both agents. Progress can only be made if the agents select complimentary actions; for example, pulling in opposite directions or attempting to rotate and push at the same time will have no effect on the overall position of the furniture piece. The *communicate* action has no special penalty associated with it, but it does not allow any other action to be activated during that cycle. Since the communicating agent remains stationary for that turn, it prevents any movement regardless of the action taken by the other agent (we assume the couch is too heavy for one agent to move on its own). All movements of the robots are discretized, and the domain can be completed optimally in 39 steps.

The staircase poses a special challenge in this domain, as it requires explicit coordination between the agents. Both agents must select the *stair* action to navigate over the stair segment successfully. However, the corridor is narrow, and the agents are forced to move one after the other instead of side-by-side. As a result, the rear agent is not able to sense when the front agent reaches the staircase. To successfully pass through this region, the front agent must communicate its stair data in order for the rear agent to recognize that the *stair* action is required. Similarly, once the front agent moves past the stairs, the rear agent must communicate its stair information to ensure that the front agent knows to continue executing the *stair* action.

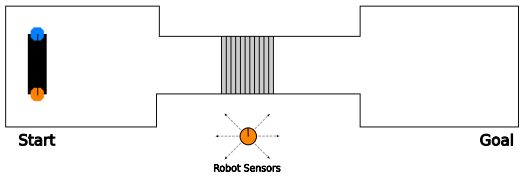


Figure 6. Screenshot of the furniture movers domain. Two agents must collaborate to move a couch from one room to another through a narrow hallway with stairs.

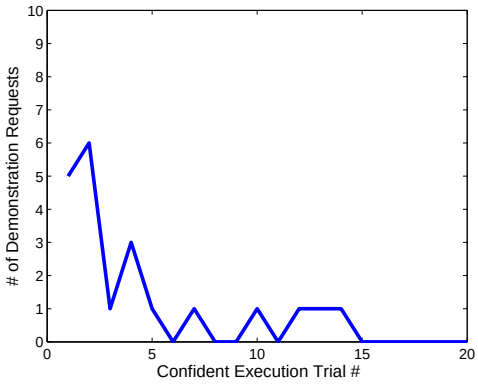


Figure 7. Total number of demonstration requests made by the agents during each trial of the confident execution training phase.

Since a single agent can not perform the task alone, both agents were trained to perform the task at the same time. Demonstrations were performed on an individual basis for each agent. During the confident execution stage, an agent requesting a demonstration waited for the teacher's response, while the other agent was free to continue its execution of the task. Note that in this task, the second agent is not able to make progress on its own due to the constraints of the domain, however, the algorithm places no restrictions upon this agent's actions.

We first evaluate the performance of our learning method using only the non-interactive demonstration technique, in which the agents have no autonomy and the expert performs exhaustive demonstrations of the task. We then present results using the complete confident execution framework. This comparison allows us to evaluate confident execution independently in the context of imitation learning.

Using only the non-interactive demonstration technique, the agents required four demonstrations of the complete domain, or a total of 156 examples per agent, to achieve 100% prediction accuracy and learn the optimal policy. This result confirms that learning from demonstration allows the agents to imitate the behavior of the expert from a small number of examples. Each agent learned its own, unique, policy; the final learned model for each agent consisted of six 8-dimensional Gaussian mixture models.

Confident execution was used to reduce the number of required demonstrations even further by eliminating demonstrations

of already acquired behavior. Training was performed using non-interactive demonstration until both models reached 80% prediction accuracy over a window of 15 timesteps, resulting in a total of 65 demonstrations per agent. Under confident execution, the agents continued to perform the task, requesting assistance from the expert at times of uncertainty. Figure 7 shows the total number of demonstration requests made by both agents during each confident execution trial. The number of demonstration requests made decreases with training, until no further requests are made after the 14th learning trial. This resulted in an overall total of 86 demonstrations per agent, approximately half of the number of demonstrations required by the non-interactive method.

Finally, we compare the performance of our algorithm to reinforcement learning. Specifically, Q-learning with a non-deterministic update function was used the learn a policy for each agent. To simplify the task, all action combinations that did not have an effect (such as one agent moving forward, while the other moves back) were not taken into account. This approach was able to learn the optimal policy after 470 iterations, and a total of 58370 exploration steps. Table 2 summarizes the results of all three learning approaches. Note that reinforcement learning performs poorly in this domain because the state of the world is not fully observable as each agent does not know the action taken by its teammate. Partial observability makes this a very challenging problem [22], and a number of special approaches have been developed for dealing with this case [24]. We plan to evaluate and compare these approaches in future work.

Algorithm	# Steps to Learn
Non-Interactive Demonstration	156
Confident Execution	86
Reinforcement Learning	58370

Table 2. Comparison of the number of cycles required to learn the optimal policy in the furniture movers domain.

5 Conclusion

In this paper, we proposed imitation as an alternative to exploration-based methods for learning in multiagent systems. We demonstrated the effectiveness of this approach using our demonstration-based learning framework in a complex simulated multiagent domain. Using our technique, we were able to quickly and accurately train the agents to imitate a human demonstration of the task. Additionally, our results showed that the confident execution approach effectively reduces the workload of the expert, allowing training to scale to a greater number of agents.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng, 'Apprenticeship learning via inverse reinforcement learning', in *International Conference on Machine learning*, New York, NY, USA, (2004). ACM Press.

[2] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn, 'Synchrony and perception in robotic imitation across embodiments', in *IEEE International Symposium on Computatonal Intelligence in Robotics and Automation*, Kobe, Japan, (2003).

[3] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn, 'Towards robot cultures?', *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, **5**(1), 3–44, (2004).

[4] R.C. Arkin, *Behavior-based robotics*, MIT Press, 1998.

[5] D. C. Bentivegna, C. G. Atkeson, and G. Cheng, 'Learning from observation and practice using primitives', *AAAI Fall Symposium Series, 'Symposium on Real-life Reinforcement Learning'*, (2004).

[6] D. C. Bentivegna, G. Cheng, and C. G. Atkeson, 'Learning from observation and from practice using behavioral primitives', *11th International Symposium of Robotics Research*, (2003).

[7] D. C. Bentivegna, A. Ude, C. G. Atkeson, and G. Cheng, 'Learning to act from observation and practice', *International Journal of Humanoid Robotics*, **1**(4), (2004).

[8] C. Breazeal, G. Hoffman, and A. Lockerd, 'Teaching and working with robots as a collaboration', in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1030–1037, Washington, DC, USA, (2004). IEEE Computer Society.

[9] Georgios Chalkiadakis and Craig Boutilier, 'Coordination in multiagent reinforcement learning: a bayesian approach', in *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 709–716, New York, NY, USA, (2003). ACM Press.

[10] S. Chernova and M. Veloso, 'Confidence-based policy learning from demonstration using gaussian mixture models', in *Joint Conference on Autonomous Agents and Multi-Agent Systems*, (2007).

[11] Jeffery Allen Clouse, *On integrating apprentice learning and reinforcement learning*, Ph.D. dissertation, University of Massachisetts, Department of Computer Science, 1996. Director-Paul E. Utgoff.

[12] A.P. Dempster, N.M.Laird, and D.B. Rubin, 'Maximum likelihood from incomplete data via the em algorithm', *Journal of Royal Statistical Society*, **8**(1), (1977).

[13] T. Inamura, M. Inaba, and H. Inoue, 'Acquisition of probabilistic behavior decision model based on the interactive teaching method', in *Ninth International Conference on Advanced Robotics (ICAR)*, pp. 523–528, (1999).

[14] L.P. Kaelbling, M.L. Littman, and A.W. Moore, 'Reinforcement learning: A survey', *Journal of Artificial Intelligence Research*, **4**, 237–285, (1996).

[15] A. Lockerd and C. Breazeal, 'Tutelage and socially guided robot learning', in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2004).

[16] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[17] M. N. Nicolescu and M. J. Mataric, 'Learning and interacting in human-robot domains', in *IEEE Transaction on Systems, Man and Cybernetics*, pp. 419–430, (2001).

[18] M. N. Nicolescu and M. J. Mataric, 'Natural methods for robot task learning: instructive demonstrations, generalization and practice', in *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 241–248, New York, NY, USA, (2003). ACM Press.

[19] Eugnio Oliveira and Luis Nunes, *Learning by exchanging Advice*, Springer, 2004.

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

[21] Bob Price and Craig Boutilier, 'Accelerating reinforcement learning through implicit imitation.', *J. Artif. Intell. Res. (JAIR)*, **19**, 569–629, (2003).

[22] D. Pynadath and M. Tambe, 'Multiagent teamwork: Analyzing the optimality and complexity of key theories and models', in *1st Conference on Autonomous Agents and Multiagent Systems*, (2002).

[23] Patrick Riley, *Coaching: Learning and Using Environment and Agent Models for Advice*, Ph.D. dissertation, Computer Science Dept., Carnegie Mellon University, 2005. CMU-CS-05-100.

[24] Maayan Roth, Reid Simmons, and Manuela Veloso, 'Reasoning about joint beliefs for execution-time communication decisions', in *The Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, (2005).

[25] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, 'Teaching robots by moulding behavior and scaffolding the environment', in *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pp. 118–125, New York, NY, USA, (2006). ACM Press.

[26] P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world, 2003.

[27] M. Sierhuis, J. Bradshaw, A. Acquisti, R. Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice, 2003.

[28] W. D. Smart and L. P. Kaelbling, 'Effective reinforcement learning for mobile robots', in *IEEE International Conference on Robotics and Automation*, (2002).

Echo State Network Applied to a Robot Docking Task

Xavier Dutoit and **Davy Sannen** and **Marnix Nuttin**[1]

Abstract. Reservoir Computing (RC) is a new technique which allows to use complex recurrent neural networks while keeping the training complexity low. We apply here RC as a high-level controller for a robot which has to perform a docking task. The RC method presents two main advantages. The task can be taught in a black-box approach, using only learning by imitation. The explicit dependence from situations to actions does not need to be coded. And RC requires only training simple readouts which can be guaranteed to find a local minimum.

1 INTRODUCTION

When controlling a robot, one wants the robot to be intelligent and autonomous. This means that the robot has to be able to decide actions by itself in a environment which is constantly changing. Moreover, as the sensors are not perfect, the robot has a noisy or even inconsistent perception of this environment.

In order to solve those problems, a lot of work has been done in the field of robotic control. This work can be divided in different categories (see [13] for a more complete description):

- **Reactive Control**: The robot has no memory but just makes a mapping from situations to action. This is simple to implement and fast to execute, but the number of tasks it can perform is rather limited.
- **Deliberative Control**: Here some more complex processing is involved, and the robot has a memory, so it can associate an action to a given situation with a given past. This allows to deal with more complex task, but requires more hardware and computation time.
- **Hybrid Control**: This approach is a trade-off between the two preceding techniques, and can allow to combine their advantages.
- **Behaviour-based Control**: As the name says, the robot has a set of behaviour. Depending on the situation, it can choose which behaviour to execute. This allows to be more flexible.

We intend to solve here a non-reactive task and will use deliberative control. However, deliberative architectures usually need explicit coding. We will rather use here another approach which would allow to solve the task in a more natural way, without coding the explicit dependency from situation to actions. Instead, in our approach, it is possible to train the robot by imitation.

To do so, we use Artificial Neural Networks (ANNs). They allow to process inputs in a nonlinear and adaptive way. Unlike classical approaches, there is no need to know in advance how to solve the task: neural networks can show an ability to learn by themselves, when provided a good set of examples. They can then generalize from this set of examples. Moreover, they are typically able to deal with noisy or inconsistent data (see for instance [25]). This altogether makes them very interesting for robotic applications. More precisely, we will consider ANNs in the framework of Reservoir Computing (RC).

[1] Katholieke Universiteit Leuven, Belgium,
email:{xavier.dutoit,davy.sannen,marnix.nuttin}@mech.kuleuven.be

2 RESERVOIR COMPUTING

The basic idea of RC is to input the data into a big recurrent network and then to train some simple readouts to extract useful information, while the network itself remains unchanged. RC has been introduced by [17] with the *Liquid State Machines* (LSMs), where the network consist of spiking neurons, and [10] with the *Echo State Networks* (ESNs), where the network consists of sigmoidal neurons. It can also be compared to the results from [24] when they studied the weight dynamics of a recurrent neural network and to the *Backpropagation-Decorrelation* algorithm [26].

The part of the network which is not trained can be seen as a reservoir of functions, and the output neurons as readouts that can extract the main features from this reservoir.

When the input is presented to the reservoir, it is in fact projected into a high dimensional and highly dynamic space. This is similar to a kernel method (see e.g. [6] for a review), and has the advantage over classical kernel methods that it can include time.

A great advantage of RC is then that we can apply simple readout functions to the reservoir, like linear discriminants, which are simple to train and can be guaranteed to find a global optimum in the offline case.

The power of reservoir computing has potentially no limit: any task can be solved as long as the desired features are present in the reservoir. On the other hand, a drawback is that the features have to be presents in the reservoir, which is not always the case, and it is typically hard to know in advance how to design a reservoir in order to make it capture those features. But if it manages to have those features, it requires absolutely no prior knowledge about the task to be solved, whereas with other approaches, some hard-coding of time-dependent actions has to be made.

We will here focus more particularly on ESNs. They are simpler to implement and simulate than LSMs, as they use classical (sigmoidal) neurons whereas LSMs use spiking neurons interconnected by synapses with a weight and a delay.

2.1 Applications of Reservoir Computing

RC has been applied with promising results in several domains, like:

Speech recognition In [28], an LSM has been trained to recognize spoken digits. The LSM has shown a good robustness against noise. It is interesting to see that, amongst 3 different pre-processing techniques of the sound, the most biological model, the Lyon Passive Ear[16], has lead to the best results.

Movement prediction In [5], an LSM has been trained to predict the movement of a ball with real-world images. It was able to predict the movement reliably up to 200 ms ahead. However, the results depend on a good choice of the parameters of the liquid.

The XOR problem and real liquid [9] used a real liquid excited by electric motors and whose image was recorded by a web-cam. They trained it to simulate a XOR logic gate and to distinguish between the spoken digits 'one' and 'zero' and showed good performances and good robustness against noise.

Real-time obstacle avoidance [4] used a LSM implemented in real-time to control a small robot and make it avoid obstacles. The learning was done by demonstration.

Arm control [14] used a LSM to control a robot arm in a biologically inspired way. The arm was trained to reach different target points. It was a first implementation of a closed-loop system controlled by neural microcircuits.

2.2 Learning by imitation

If we control a robot with reservoir computing, as nothing is programmed beforehand, it has to learn the task. A very appealing way is to make it learn by imitation. It consists of showing the robot a desired behaviour in order to make it learn to exhibit the same behaviour afterward, when the same situation is presented (see for instance [2] for a review).

Learning by imitation is very appealing because of its conceptual simplicity when compared to other methods. Typically, it is often much simpler to show a robot what to do by doing it ourselves than to program it. It has also the advantage that it does not necessarily require concrete knowledge about the robotic domain: a person who does not know how a robot has to be programmed can still show some tasks to the robot. This advantage is interesting, especially if we consider the application of domestic robots, where anybody could teach a robot a given task in this way. It is very appealing for cooperation between human and robot and for real-world learning applications [1], [3], [8], [18], [19], [23].

Moreover, learning by imitation is a natural way to teach and learn for human beings and animals. It is very commonly observed amongst monkeys, for instance, and in fact it is the reason of the name *aping*.

3 THE EXPERIMENT

3.1 Related work

We try here to apply the technique of RC to learn a docking task. ESNs have already been applied to control task (see for instance [21, 22, 20]). However, the previous applications generally use the ESN as a low-level controller of which the goal is to output the motor command based on a desired trajectory. In our approach, we first present a set of trajectories to learn, but then, during the testing phase, the ESN has to decide the trajectory based on the sonar input only.

The docking task has also already been solved with a behaviour-based approach [15]. However, when using a behaviour-based approach, the task needs to somehow be segmented in the different manoeuvres the robot will have to make. On the other hand, with RC, the raw data is fed to the reservoir, without any preprocessing or prior knowledge involved.

This task can also be solved using planners [27],[7]. However, we are interested here in a more adaptive and flexible approach, as we think it might exhibit some interesting features in the long run.

3.2 Goal

The robot must perform a docking task, i.e. it must first go backward and then go forward and turn left (cf. Fig. 1). It starts around the point indicated by 'Start', oriented towards the positive direction of the x axis. In one zone, the shaded area, there are some points where the robot will go twice, and thus be twice in the same situation, but with different desired outputs. So the task is not a purely reactive task, it features time dependency.

A run is considered as successful if the robot first goes back enough to enter the shaded zone (cf. Fig. 1) and then reaches the goal area.

3.3 Setup

We first simulated a robot based on the PIONEER-1 robots. It has 7 sonars placed symmetrically at its front (see Fig. 1), at the angles of ± 90,± 30,± 15 and 0 degrees (0 being the forward direction of the robot).

The robot can give discrete commands out of two independent sets:

- Linear velocity: go forward or go backward
- Rotational velocity: turn-left, do-not-turn or turn-right

A step forward or backwards corresponds to a distance of 100 mm, a turn left or right to an angle of $\frac{\pi}{8}$.

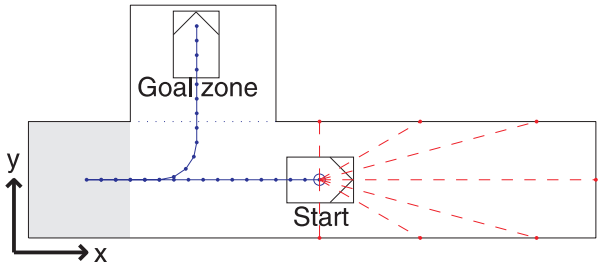


Figure 1. The Docking Task. The robot starts at the point indicated by *Start*; it moves along the solid line, each dot representing a step; the shaded area represent the contradictory zone; the dotted lines indicate the sonars for the robot at the starting position.

3.4 Training data

The network is trained by demonstration. 60 different trajectories were created by a human supervisor (20 from the original starting point, and 40 from other random points in the environment), and they were then randomly shifted to create new trajectories. In total, the training set consists of 494 trajectories.

For each trajectory, we record the 7 sonar data. In the simulation environment, those data do not get any noise. Each sonar returns the distance (in [mm]) to the nearest wall.

For each trajectory, the sonar data were fed to the network and the 3 readouts were trained to map the states of the network onto the desired motor commands (see section 4.5 for the detailed training procedure).

4 THE ECHO STATE NETWORK

The ESN considered here consists of one input layer, one reservoir, and one output layer.

There are n_i neurons in the input layer, n_r neurons in the reservoir and n_o neurons in the output layer.

In the present task, $n_i = 7$ (the 7 sonars inputs), and $n_o = 3$ (we use 3 outputs to command the two velocities, the mapping from outputs to commands is described below, section 4.3)

4.1 Input

At each time step t, the input vector $\mathbf{i}(t)$ is multiplied by a input weight matrix $\mathbf{W}_I$, of size $n_r \times n_i$, and fed to the reservoir.

4.2 Reservoir

The reservoir, consisting of n_r neurons, is described by a connection matrix $\mathbf{W}$, of size $n_r \times n_r$, and at each time step by a state vector $\mathbf{s}(t)$. This state vector is all zero at the beginning and is updated according to the following equation:

$$\mathbf{s}(t+1) = f\Big(m\cdot\big(\mathbf{W}_I\cdot\mathbf{i}(t)+\mathbf{W}\cdot\mathbf{s}(t)\big)+(1-m)\cdot\mathbf{s}(t)\Big) \quad \forall\, t>0 \quad (1)$$

$$\mathbf{s}(0) = 0$$

where f can be any linear or non-linear function (here we use a sigmoidal function, the hyperbolic tangent), and m ($0 \leq m \leq 1$) is a parameter tuning the dynamic of the reservoir.

4.3 Output

Each readout r is a linear discriminant, described by a weight vector $\mathbf{W}_r$. The output of the network O_r at time t is given by:

$$O_r(t) = \mathbf{W}_r \cdot \bar{\mathbf{s}}(t) \quad (2)$$

where $\bar{\mathbf{s}}(t)$ is the state vector augmented with a bias term:

$$\bar{\mathbf{s}}(t) = \begin{bmatrix} \mathbf{s}(t) \\ 1 \end{bmatrix}$$

In our case, there are 3 readouts, one for the linear velocity V, two for the rotational velocity R. The actual commands are:

$$V(t) = \begin{cases} +1 & \text{if } O_1(t) > 0 \\ -1 & \text{otherwise} \end{cases} \quad (3)$$

$$R(t) = \begin{cases} +1 & \text{if } O_2(t) - O_3(t) > \Theta \\ -1 & \text{if } O_3(t) - O_2(t) > \Theta \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Θ being a threshold factor, determined experimentally.

4.4 Network creation

The reservoir is created at random, according to the following parameters:

n_i	the number of inputs
n_r	the size of the reservoir
c_i	the input connection fraction
c_r	the reservoir connection fraction
i_w	the weights distribution of the inputs
m	the memory parameter

The input connection matrix $\mathbf{I}$ is a $n_r \times n_i$ matrix with a proportion c_i of non-zero weights. Those non-zero weights take on their values uniformly in i_w (in our case, $i_w = \{-0.1; +0.1\}$).

The reservoir connection matrix $\mathbf{C}$ is a $n_r \times n_r$ matrix with a proportion c_r of non-zero weights. Those non-zero weights take on their values out of a 0-mean gaussian distribution with variance 1.

Once generated, this connection matrix is rescaled: it is divided by its spectral radius (so that its spectral radius is 1 after rescaling). This rescaling allows to stand at the limit of the *echo state property* [10].

4.4.1 Effect of the memory parameter m

The parameter m allows to have leaky neurons, i.e. neurons which have a certain memory. Indeed, if $m < 1$, at each time step, a neuron will have as net input (i.e. before applying the non-linear function) the net input from other neurons multiplied by m and $(1-m)$ times its own delayed input. In the absence of external input, the activity level of a given neuron exponentially decays with a time constant of $\frac{1}{m}$[time steps].

Now concerning the echo state property, one has to note that we can rewrite equation 1 as:

$$\mathbf{s}(t+1) = f\left(\widetilde{\mathbf{W}}_I \cdot \mathbf{i}(t) + \widetilde{\mathbf{W}} \cdot \mathbf{s}(t)\right) \quad \forall\, t>0$$

where $\widetilde{\mathbf{W}}_I = m \cdot \mathbf{W}_I$ and $\widetilde{\mathbf{W}} = m \cdot \mathbf{W} + (1-m) \cdot \mathbf{I}$ ($\mathbf{I}$ being the identity matrix).

As $\mathbf{W}$ has a spectral radius equal to one, i.e. all its eigenvalues are smaller or equal to one, $\widetilde{\mathbf{W}}$ has all its eigenvalues smaller or equal to m, and its spectral radius equal to m. So the echo state property is guaranteed for $m < 1$, and we stand at the limit of this property when $m = 1$.

4.5 Training

The training set consist of a set of n_t vector of size n_i, and of a set of n_t associated desired output pairs $(\hat{\mathcal{V}}(t), \hat{\mathcal{R}}(t))$. For each sample t, we define the 3 desired output $(\hat{O}_1(t), \hat{O}_2(t), \hat{O}_3(t))$ as follows:

$$\hat{O}_1(t) = \hat{\mathcal{V}}(t)$$

$$\hat{O}_2(t) = \begin{cases} +1 & \text{if } \hat{R}(t) = +1 \\ -1 & \text{otherwise} \end{cases}$$

$$\hat{O}_3(t) = \begin{cases} +1 & \text{if } \hat{\mathcal{R}}(t) = -1 \\ -1 & \text{otherwise} \end{cases}$$

Now to do the actual training, the network is fed with the n_t input samples, and we collect the augmented states in a matrix $\mathbf{S}$:

$$\mathbf{S} = [\bar{\mathbf{s}}(1)\, \bar{\mathbf{s}}(2)\, \ldots\, \bar{\mathbf{s}}(n_t)]$$

The readouts are computed by solving the following equation in the least square sense:

$$\mathbf{W}_r \cdot \mathbf{S} = \hat{\mathbf{O}}_r \qquad r = 1, 2, 3$$

where $\hat{\mathbf{O}}_r$ is the vector containing the desired output for all the n_t samples:

$$\hat{\mathbf{O}}_r = [\hat{O}_r(1)\ \hat{O}_r(2)\ \ldots\ \hat{O}_r(n_t)]$$

The actual commands are then computed according to (2), (3) and (4).

4.6 Training error

The training error is the proportion of wrong commands over the training set, defined as:

$$E_T = \sum_t e(t)$$

where:

$$e(t) = \begin{cases} 0 & \text{if } V(t) = \hat{V}(t) \text{ and } R(t) = \hat{R}(t) \\ 1 & \text{if } V(t) \neq \hat{V}(t) \text{ and } R(t) \neq \hat{R}(t) \\ 0.5 & \text{otherwise} \end{cases}$$

4.7 Testing error

Once the network was trained, it was tested starting from 10 different point chosen randomly around the original starting point according to a normal distribution of mean 0 and of variance 200 mm on the x axis and 100 mm on the y axis. To avoid the robot starting too close to a wall or outside the world, the starting point was limited to be no more than 500 mm and 300 mm away from the starting point, on the x and y axis resp.

The testing error is the proportion of trajectories which did not fulfill the success criterion (see above, section 3.2).

5 RESULTS

We applied here an ESN approach to teach a robot to perform a docking task. Several reservoirs were created randomly, without any programming of the task beforehand, and were trained by demonstration to reproduce the training runs. By training only 3 linear discriminants, it is possible to achieve an average success rate of 76 % on testing (see Fig. 2 for examples of successful trajectories). Some of the networks managed to perform the task successfully in all the cases tested.

In the present experiment, as the task is time-dependent, an important point is the memory of the reservoir. So far, there exist little methodology or measure of the memory of a given reservoir[11, 12]. However, we can say that the memory roughly depends on two parameters: the reservoir size n_r and the memory scale m.

The n_r controls the memory on a global scale. When all parameters stay constant, a bigger reservoir will mean that there exist potentially longer loops inside the reservoir, and the input will thus have longer echoes.

On the other hand, m controls the memory on a local scale: the smaller m is, the longer is the memory of a given neuron, as a past input will have an exponentially decreasing effect for a longer time. However, m also scales down the global spectral radius, thus changing the memory on a global scale as well.

The general results for those two parameters are shown in Fig. 3.

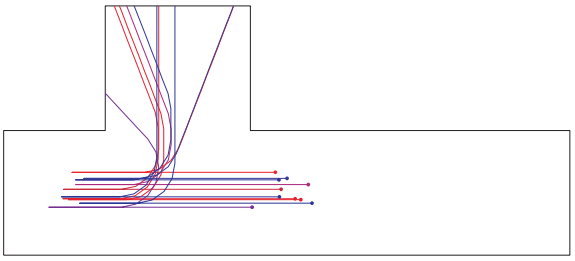


Figure 2. Example of trajectories with different starting points around the original starting point

5.1 Reservoir size

If we take a closer look at the effect on n_r (see Fig. 4), we see that in the present experiment a bigger reservoir leads to better results. This is because it allows to have more memory, but also because a bigger reservoir is likely to capture more features from the input, and thus is more likely to capture the relevant features for the task.

5.2 Memory scale

If we now look at the effect of m, the memory scale, we see that a smaller m, corresponding to a longer memory, produces on average better results. However, for the testing error, there is a lower limit under which the test error starts to increase again.

One can notice than even with a badly scaled memory, we can still have around 40 % of the networks which succeed to perform the task. This shows that it is not required to know in advance what are the memory requirements of the task in order to be able to perform it successfully.

There is also a second noticeable point: in the training samples, the time spent in the contradictory input zone (shaded area in Fig. 1) was around 7 steps. So the robot saw twice the roughly same input, first at a given time and then 7 time steps later. So we can roughly say that the memory requirements for this task is about 7 time steps, i.e. a robot must have a memory spanning at least 7 time steps in order to perform the task. But we can see that with $m = 0.25$, i.e. when each neuron has a time scale of 4 steps, there were still around 35 % of the networks which performed the task successfully. So even when the memory is badly scaled, it is possible to sometimes succeed in performing the task. This shows that a reservoir can exhibit on a global scale a behaviour on a time scale larger than the local time scale of any of its element.

5.3 Extension to more realistic environment

We then applied the task in the environment of the Saphira robot simulator. This means that there was some noise in the input and in the output, i.e. the sonars data were noisy and the commands were not perfectly executed. Moreover, in this environment, the robot has now a radius of about 200 mm and so it has less margin to manoeuver. So the sonars data were all getting subtracted 200 before being fed to the robot.

It succeeded both in the simulation environment and with a real robot (Fig. 7), and an example of successful trajectories is shown in Fig. 6.

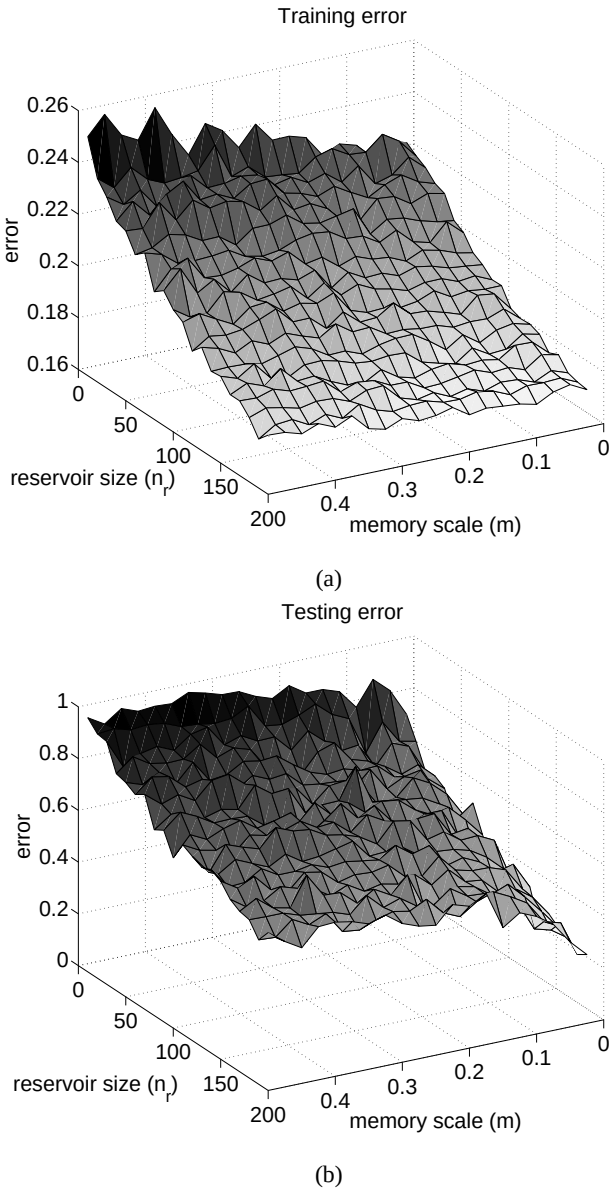


Figure 3. General view of the effect of n_r and m: (a) training error, (b) testing error

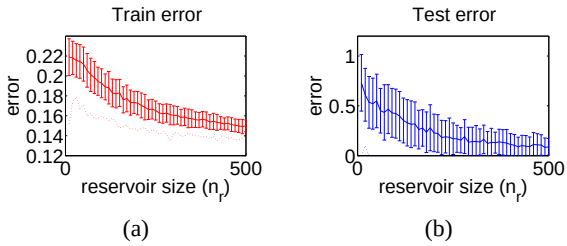


Figure 4. Effect of the reservoir size, n_r: (a) training error, (b) testing error. The solid line represents the average error with the standard deviation, the dotted line represents the minimum error.(out of 100 simulations, $m = 0.02$)

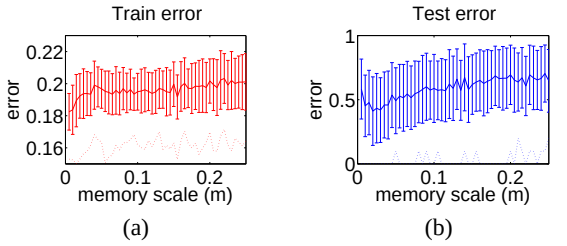


Figure 5. Effect of the memory scale, m: (a) training error, (b) testing error. The solid line represents the average error with the standard deviation, the dotted line represents the minimum error.(out of 100 simulations, $n_r = 100$)

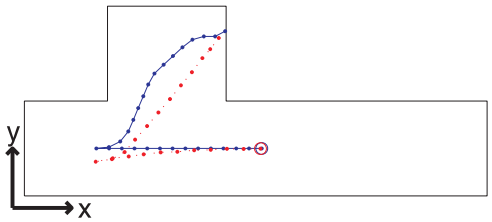


Figure 6. Example of the application in the Saphira environment: solid line: simulation environment, dotted line: real world.

6 CONCLUSION

We considered here an application of an emerging technique: reservoir computing. To the best of our knowledge, it is the first time RC is applied on a high-level to a navigation problem such as this docking task. With RC, it has been possible to successfully perform a control task in the simulated world, as well as in the real world, where the sensor readings were noisy, and the commands were not perfectly executed.

This RC technique allows a simple training. Indeed, it is sufficient to generate some training examples and to show them to the network. Thus we do not need to know the explicit correspondence from input to output, which is typically hard to know. Moreover, we can also use a very simple training algorithm, which is guaranteed to find an optimal solution (in the least-square sense): the training consists of a matrix inversion, which can be computationally expensive, but is straightforward and guarantees to find the global optimum. A drawback is that this method can only be implemented offline. The RC approach also allows to be flexible, and even though some parameters have to be tuned and tested, it is possible to perform the desired task even with badly scaled parameters. Thus we think that this technique is promising for robot task control.

ACKNOWLEDGEMENTS

This work was partially funded by FWO Flanders project G.0317.05, and by the Belgian government under the Inter-University Attraction Poles, Office of the Prime Minister, IAP-AMS.

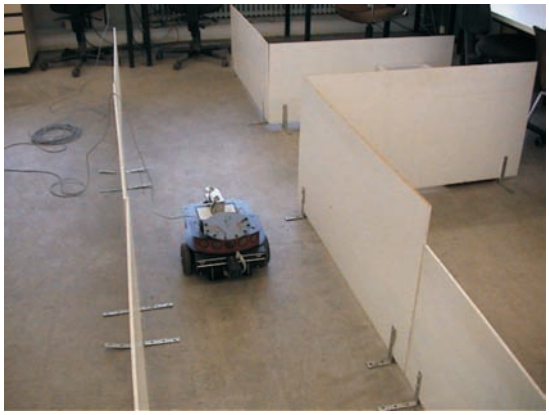


Figure 7. The real world setup

REFERENCES

[1] P. Baker and Y. Kuniyoshi, 'Robot see, robot do: an overview of robot imitation', *AISB96 Workshop on Learning in Robots and Animals*, 3–11, (1996).

[2] A. Billard and R. Siegwart, 'Robot learning by demonstration', *Robotics and Autonomous Systems*, **47**(2-3), 65–67, (2004).

[3] E. Burdet and M. Nuttin, 'Learning complex tasks using a stepwise approach', *Journal of Intelligent and Robotic Systems*, **24**, 43–69, (1999).

[4] H. Burgsteiner, 'Training networks of biological realistic spiking neurons for real-time robot control', *Proceedings of the EANN 2005*.

[5] H. Burgsteiner, M. Kröll, A. Leopold, and G. Steinbauer, 'Movement prediction from real-world images using a liquid state machine', in *Proceedings of the 18th Internal Conference IEA/AIE*. Springer-Verlag, Berlin, Germany, (2005).

[6] C. Campbell, 'Kernel methods: a survey of current techniques', *Neurocomputing*, **48**, 63–84, (2002).

[7] E. Demeester, M. Nuttin, D. Vanhooydonck, and H. Van Brussel, 'Fine motion planning for shared wheelchair control: Requirements and preliminary experiments', *Int. Conf. on Adv. Robotics*, 1278–1283, (2003).

[8] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zollner, and M. Bordegoni, 'Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm', *9th International Symposium of Robotics Research*, (1999).

[9] C. Fernando and S. Sojakka, 'Pattern recognition in a bucket', in *Proc. of the ECAL 2003*, (1998).

[10] H. Jaeger, 'The "echo state" approach to analysing and training reccurent neural networks', Technical Report GMD 148, German National Research Center for Information Technology, (2001).

[11] H. Jaeger, 'Short-term memory in echo state networks', Technical report, (2002).

[12] H. Jaeger, 'A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach', Technical report, (2002).

[13] C. Jones and M. Matarić, 'Behavior-based coordination in multi-robot systems', *Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications*, (2005).

[14] P. Joshi and W. Maass, 'Movement generation and control with generic neural microcircuits', *Proceedings of BIO-ADIT*, (2004).

[15] M. Kasper, G. Fricke, K. Steurenagel, and E. von Puttkamer, 'A behavior-based mobile robot architecture for learning from demonstration', in *Robotics and Autonomous Systems*, volume 34, pp. 153–164, (2001).

[16] R.F. Lyon, 'A computational model of filtering, detection and compression in the cochlea', in *Proc. IEEE Int. Conf. Acoustics Speech and Signal Processing*, (1982).

[17] W. Maass, T. Natschläger, and H. Markram, 'Real-time computing without stable states: A new framework for neural computation based on perturbations', *Neural Computation*, **14**(11), 2531–2560, (2002).

[18] M. Nuttin, *Learning Approaches to Robotics Manufacturing: Contributions and Experiments*, Ph.D. dissertation, K.U.Leuven, 1998.

[19] M. Nuttin and H. Van Brussel, 'Learning assembly operations: A case study with real-world objects', *Studies in Informatics and Control*, **3**, 205–221, (1996).

[20] M. Oubbatti, *Neural Dynamics for Mobile Robot Adaptive Control*, Ph.D. dissertation, Universität Stuttgart, 2006.

[21] P.G. Ploeger, A. Arghir, T. Günther, and R. Hosseiny, 'Echo state networks for mobile robot modeling and control', *Proceedings of the ROBOCUP 2003*.

[22] M. Salmen and P.G. Ploeger, 'Echo state networks used for motor control', *Proceedings of the ICRA 2005*.

[23] S. Schaal, A. Ijspeert, and A. Billard, 'Computational approaches to motor learning by imitation', *Philosophical Transactions of the Royal Society B: Biological Sciences*, **358**(1431), 537–547, (2003).

[24] U.D. Schiller and J.J. Steil, 'On the weights dynamics of reccurent learning', in *ESANN'2003 Proceedings*, pp. 73–78, (2003).

[25] J.W. Shavlik, R.J. Mooney, and G.G. Towel, 'Symbolic and neural learning algorithms: An experimental comparison', *Machine Learning*, **6**(2), 111–144, (1991).

[26] J.J. Steil, 'Backpropagation-decorrelation: online reccurent learning with o(n) complexity', in *Proc. IJCNN*, volume 1, pp. 843–848, (2004).

[27] J. Vandorpe, *Navigation techniques for the mobile robot LiAS*, Ph.D. dissertation, K.U.Leuven, 1997.

[28] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, 'Isolated word recognition with the liquid state machine: a case study', *preprint submitted to Elsevier Science*, (2005).

Can Motionese Tell Infants and Robots "What to Imitate"?

Yukie Nagai [1] and **Katharina J. Rohlfing** [2]

Abstract. An open question in imitating actions by infants and robots is how they know "what to imitate." We suggest that parental modifications in their actions, called *motionese*, can help infants and robots to detect the meaningful structure of the actions. Parents tend to modify their infant-directed actions, e.g., put longer pauses between actions and exaggerate actions, which are assumed to help infants to understand the meaning and the structure of the actions. To investigate how such modifications contribute to the infants' understanding of the actions, we analyzed parental actions from an infant-like viewpoint by applying a model of saliency-based visual attention. Our model of an infant-like viewpoint does not suppose any a priori knowledge about actions or objects used in the actions, or any specific capability to detect a parent's face or his/her hands. Instead, it is able to detect and gaze at salient locations, which are standing out from the surroundings because of the primitive visual features, in a scene. The model thus demonstrates what low-level aspects of parental actions are highlighted in their action sequences and could attract the attention of young infants and robots. Our quantitative analysis revealed that motionese can help them (1) to receive immediate social feedback on the actions, (2) to detect the initial and goal states of the actions, and (3) to look at the static features of the objects used in the actions. We discuss these results addressing the issue of "what to imitate."

1 INTRODUCTION

Imitation learning is a promising approach for robotics researchers to enable their robots to autonomously acquire new skills from humans [21, 31]. It allows robots to learn new behaviors by first observing human movements and then reproducing them by mapping into their motor commands. It consequently reduces the efforts of designers in developing robots' behaviors. In addition to these engineering benefits, the research on imitation learning leads us to the deeper understanding of human intelligence [2]. Human infants, even neonate [25, 26], are able to imitate actions. In the course of their development, infants can reproduce actions and the goal of actions shown by another person. The ability to imitate is moreover discussed as a route to their further cognitive development, e.g., the differentiation of the self and other, the understanding of other's intention, and the use of language [9]. Thus, to investigate the mechanism for imitation learning from a constructivist viewpoint allows us to uncover human intelligence [2].

There are some advantages in robot imitation, however, we still have an open question of how robots know "what to imitate" and "how to imitate." Nehaniv and Dautenhahn [28, 29] discussed these two fundamental issues in robot imitation. Breazeal and Scassellati [7, 8] also pointed out the issues and reported the current techniques used in robot systems. When a robot attempts to imitate a human action or a sequence of his/her actions to achieve a goal-oriented task, it has to first detect the movements of the person and then determine which movements are relevant to the task. A robot without any a priori knowledge about the task does not know which actions of the person are important and necessary for the task, while he/she sometimes produces not only actions directly related to the task but also unrelated ones. It is also required to detect the initial and goal states of the actions and the objects involved in the actions so that a robot can imitate the sequence of the actions not only at a trajectory level but also at a goal level. These problems are stated as the issue of "what to imitate," and several approaches have been proposed from different perspectives (e.g., [4, 6, 10, 11, 34]).

Another issue to be solved in robot imitation is how a robot knows "how to imitate." A robot that tries to imitate human actions has to be able to transform the observed actions of a person into its motor commands so as to reproduce the same actions or to achieve the same goal of the actions. A difficulty in transforming the actions is that a robot cannot access to the somatosensory information of the person and is thereby unable to directly map the actions into the motor commands. Moreover, the body structure of a robot is usually different from the person's, which makes the problem more difficult. These issues are called "how to imitate" and have been investigated from various approaches (e.g., [1, 3, 4, 10]).

In addressing these issues from a standpoint of cognitive developmental robotics [2], we suggest that parental modifications in their infant-directed actions can help robots as well as infants to imitate the actions [12, 30]. When infants attempt to imitate actions presented by their parents, they also face the same problems: "what to imitate" and "how to imitate." Although infants are supposed to have little semantic knowledge about actions as robots do, they are surprisingly able to imitate the actions. They are skillful in processing a stream of ongoing activity into meaningful actions and organizing the individual actions around ultimate goals [33]. We thus consider that parental actions aid infants solving "what to imitate" and "how to imitate." It is known that parents tend to modify their actions when interacting with their infants (e.g., [5, 30]). They, for example, put longer and more pauses between their movements, repeat the same movements, and exaggerate their movements when interacting with infants compared to when interacting with adults. Such modifications, called *motionese*, are suggested to aid infants structuring the actions and understanding the meaning of the actions. However, we do not know yet how it actually affects and contributes to the infants' understanding of the actions. Because the current researches have analyzed motionese only from an adult's viewpoint, i.e., they focused

[1] Bielefeld University, Germany, email: yukie@techfak.uni-bielefeld.de
[2] rohlfing@techfak.uni-bielefeld.de

only on the actions relevant to a task, it is still unclear what aspects of parental actions would be attended to by infants and how they help infants to understand and imitate the actions.

We analyze motionese from an infant-like viewpoint and discuss how it can help infants and robots to detect "what to imitate." Our model of an infant-like viewpoint does not suppose any a priori knowledge about actions or objects used in the actions. It does not know which parental actions are relevant to a task, what the goal of the task is, or what objects are involved in the task. Furthermore, it is not equipped with any specific ability to detect a parent's face or his/her hands. Instead, it is able to detect and gaze at outstanding locations in a scene. To simulate such a capability of visual attention, we adopt a model of saliency-based visual attention [16, 17] inspired by the behaviors and the neural mechanism of primates. A salient location in this model is defined as a location which locally stands out from the surroundings because of its color, intensity, orientation, flicker, and motion [16]. It thus can demonstrate what low-level aspects of parental actions are highlighted in their action sequences and could attract the attention of young infants and robots. We analyze motionese with the model and discuss the results toward solving the issue of "what to imitate."

The rest of this paper is organized as follows. In Section 2, we summarize the current evidences of motionese from psychological and computational studies. In Section 3, we introduce the model of saliency-based visual attention and describe the benefits of using it for the analysis of motionese. Next, we show analytical experiments of motionese in Section 4, and discuss the experimental results in Section 5. Finally, we conclude with future directions in Section 6.

2 PARENTAL MODIFICATIONS IN INFANT-DIRECTED INTERACTIONS

It is well known that parents significantly alter the acoustic characteristics of their speech when talking to infants (e.g., [19]). They, for example, raise the overall pitch of their voice, use wider pitch, slow the tempo, and increase the stress. These phenomena, called *motherese*, are suggested to have the effects of attracting the attention of infants and providing easily structured sentences to infants, which consequently facilitates their language learning.

In contrast to motherese, motionese is phenomena of parental modifications in their actions. Parents tend to modify their actions when interacting with infants so that they maintain the attention of infants and highlight the structure and the meaning of the actions as in motherese. Brand et al. [5] revealed that mothers altered their actions when demonstrating the usage of novel objects to their infants. They videotaped mothers' interactions first with an infant and then with an adult, and manually coded them on eight dimensions: the proximity to the partner, the interactiveness, the enthusiasm, the range of the motion, the repetitiveness, the simplification, the punctuation, and the rate. Their results comparing the infant-directed interactions (IDI) and adult-directed interactions (ADI) revealed significant differences in the first six dimensions out of the eight (higher rates in IDI than in ADI). Masataka [22] focused on a signed language and found that deaf mothers also altered their signed language. He observed deaf mothers when interacting with their deaf infants and when interacting with their deaf adult friends, and analyzed the characteristics of their signs. His comparison indicated that, when interacting with infants, deaf mothers significantly slowed the tempo of signs, frequently repeated the same signs, and exaggerated each sign. His further experiments showed that such modifications in a signed language attracted greater attention of both deaf and hearing infants [23, 24]. Gogate et al. [14] investigated the relationship between maternal gestures and speech in a object-naming task. They asked mothers to teach their infants novel words by using distinct objects and observed how the mothers used their gestures along with their speech. Their results showed that mothers used the target words more often than non-target words in temporal synchrony with the motion of the objects. They thus suggested that maternal gestures likely highlighted the relationship between target words and objects, of which effects were demonstrated in their further experiment [13]. Iverson et al. [18] also revealed that maternal gestures tended to co-occur with speech, to refer to the immediate context, and to reinforce the message conveyed in speech in daily mother-infant interactions. Their analysis moreover showed positive relationships between the production of maternal gestures and the verbal and gestural productions and the vocabulary size of infants.

In contrast to the former studies, in which motionese was manually coded, Rohlfing and her colleagues [12, 30] applied a computational technique to evaluate motionese. They adopted a 3D body tracking system [32], which was originally developed for human-robot interactions, to detect the trajectory of a parent's hand when he/she was demonstrating a stacking-cups task to his/her infant first and then to an adult. Their quantitative analysis revealed that parents put longer and more pauses between actions and decomposed a rounded movement into several linear movements in IDI compared with in ADI. They suggested with these results that motionese can help infants and robots to detect the meaning of actions. This approach is very attractive for robotics researchers because their model can be immediately implemented into robots and enables them to leverage the advantages of motionese in imitation learning. However, it is still an open question how robots know "what to imitate." Although their study as well as the former studies showed that parents modify their task-relevant actions so as to be easily understood, robots as well as young infants do not know which parents' actions are relevant to a task. To address this problem, we apply a model of saliency-based visual attention to the analysis of motionese.

3 SALIENCY-BASED VISUAL ATTENTION

3.1 Architecture of model

To analyze motionese from an infant-like viewpoint, i.e., without any a priori knowledge about actions or objects used in the actions, we adopt a model of saliency-based visual attention [16, 17]. The model, inspired by the behavior and the neuronal mechanism of primates, can simulate the attention shift of humans when they see natural scenes. Humans are able to rapidly detect and gaze at salient locations in their views. A salient location here is defined as a location which locally stands out from the surroundings because of its color, intensity, orientation, flicker, and motion [16]. For example, when we see a white ball in a green field, we can rapidly detect and look at the ball because of its outstanding color, intensity, and orientation. When a dot is moving left while a number of dots moving right, the former dot will be tracked visually because of its distinguished motion. The model of saliency-based visual attention imitates such a primal but adaptable attention mechanism of humans.

Figure 1 shows the overview of the model used in our experiment. This is the same as the model proposed in [16] excepting the absence of the mechanism of "inhibition of return," which inhibits the saliency of locations that have been gazed at. It means that our model determines attended locations frame by frame independently. The model works as follows:

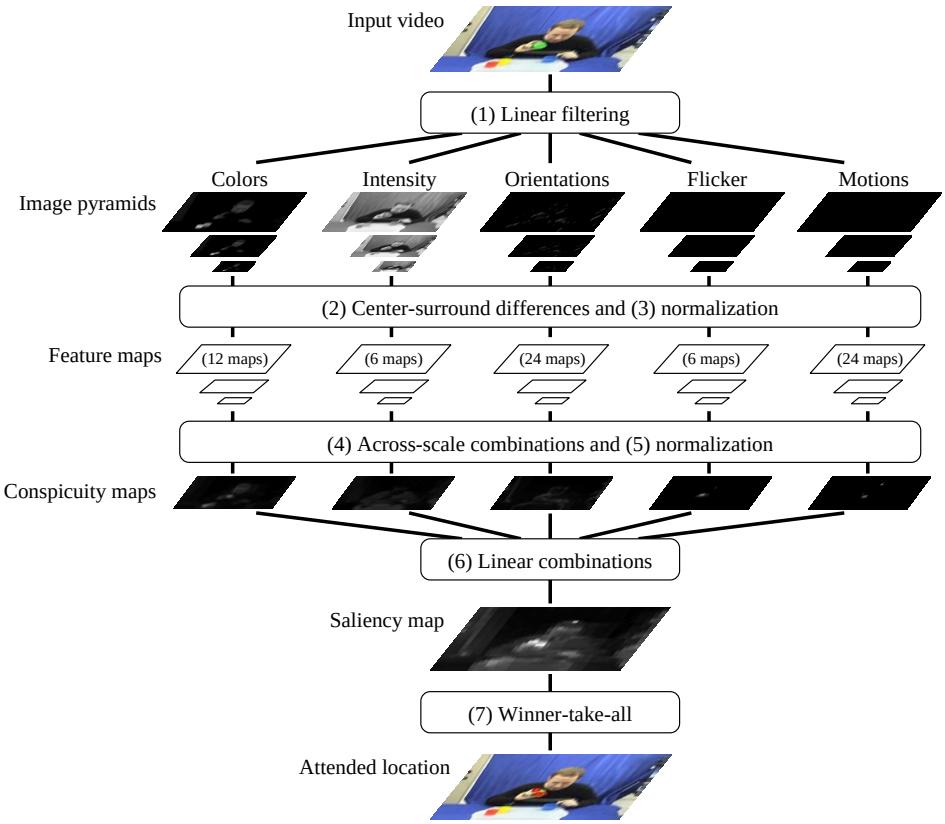


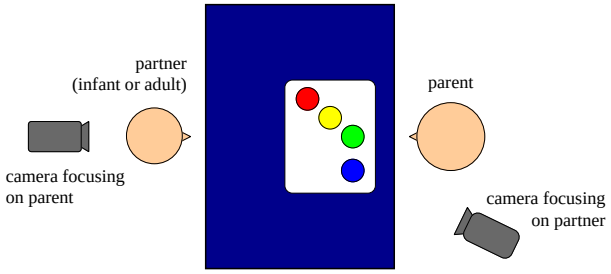
Figure 1. A model of saliency-based visual attention, which was revised from original one proposed in [17]

1. Five visual features (colors, intensity, orientations, flicker, and motions) are first extracted by linearly filtering a frame of an input video, and then image pyramids with different scales are created.
2. The differences between a center-fine scale and a surround-coarser scale image are calculated to detect how much each location stands out from the surroundings.
3. The center-surround differences are normalized to first eliminate modality-dependent differences and then globally promote maps containing a few conspicuous locations while globally suppressing maps containing numerous conspicuous peaks. The results are called feature maps.
4. The feature maps are combined through the across-scale addition to get together the different scales into one map.
5. The combined maps are normalized again to obtain conspicuity maps.
6. The conspicuity maps of the five features are linearly summed into a saliency map.
7. Finally, the most salient locations in the saliency map are selected as the attended locations in the frame.

In our analysis, image locations of which saliency were higher than the maximum $\times$ 0.9 in each frame were selected as the attended locations. That is, not only one location but several locations could be attended to in a frame. Refer to [16, 17] for more detail explanations of the processing.

3.2 Benefit of applying model to analysis of motionese

Applying the model to the analysis of motionese enables us to reveal what visual features of parental actions are highlighted in their action streams and could attract the attention of young infants and robots. Over the first year of life, infants semantic knowledge of actions, such as environmental, social, and psychological constraints on their organization and structure, is quite limited in comparison to adults. Thus, infants do not clearly understand the meaning or the structure of the actions when they see the actions for the first time. They also have limited information about objects, e.g., what objects are involved in the actions and what the initial and goal states of the objects are. Instead, they are certainly able to detect and gaze at salient locations in their views. For example, when colorful toys are shown to infants (usually, infants' toys have bright colors like yellow, red, and blue), they will look at the toys because of their salient colors. When a parent moves his/her hand to grasp and manipulate the toys, the hand as well as the toys will attract the attention of infants. Assuming only perceptual saliency, a parent's face can also attract the infants' attention because both of its static visual features and of its movement caused by his/her smiling and talking. Note that a parent's face and his/her hands can be attended to as salient locations without supposing any specific capability to detect their features or even skin color. We aim at evaluating how much meaningful structures of parental actions are detected without any knowledge about actions, objects, or humans, and how they can contribute to solving the problem of "what to imitate."



(a) top-view of experimental setup

(b) image frame of video focusing on parent, which was used as input to saliency-based attention model

(c) image frame of video focusing on infant

Figure 2. Experimental setup and sample image frames of videos

(a) input image, in which attended locations denoted by circles

(b) saliency map (sum of (c)-(g))

(c) color map (d) intensity map (e) orientation map

(f) flicker map (g) motion map

Figure 3. Example of saliency map equally summing up five conspicuity maps and attended locations

4 ANALYSIS OF MOTIONESE WITH SALIENCY-BASED ATTENTION MODEL

4.1 Method

We analyzed the videotaped data used in [30]. In contrast to [30], in which only the task-related parental actions were analyzed, we dealt with all visual features in the scenes.

4.1.1 Subjects

Subjects were 15 parents (7 fathers and 8 mothers) of preverbal infants at the age of 8 to 11 months ($M = 10.56,\ SD = 0.89$). We chose this age because infants start to imitate simple means-end actions such as acting on one object to obtain another [33] and to show the understanding of goal-directed actions at 6 months of age [20].

4.1.2 Procedure

Parents were instructed to demonstrate a stacking-cups task to an interaction partner while explaining him/her how to do it. The interaction partner was first their infants and then an adult. **Figure 2** (a) illustrates the top-view of the experimental setup, and (b) and (c) show sample image frames of cameras which were set behind a parent and a partner and focused on each of them. The stacking-cups task was to sequentially pick up the green, the yellow, and the red cups and put them into the blue one on the white tray.

4.1.3 Analysis

We analyzed videos recording the parents' actions as shown in Figure 2 (b). The videos were input to the model of saliency-based visual attention, and image locations with high saliency were detected as the attended locations frame by frame. **Figure 3** shows how the attended locations were determined in a frame: (a) shows an input image (320 $\times$ 256 [pixels]), in which three attended locations are denoted by red circles, and (b) shows the saliency map of the scene (40 $\times$ 32 [pixels]), which sums up the five conspicuity maps: (c) the color, (d) the intensity, (e) the orientation, (f) the flicker, and (g) the motion maps. The view of the maps corresponds to the input image, and the brightness of the pixels represents the degree of saliency, i.e., white means high saliency while black means low. In the example, the father was showing the green cup to his infant by shaking it, and therefore the cup and his right hand were attended to by the model. The color map extracted the green, the yellow, and the red cups as well as the father's face and hands as salient locations, while the intensity map detected the white tray and the father's black cloth. The orientation map detected the father's face, his hands, and the contour of the tray because of their rich edges. The flicker and the motion maps extracted the father's right hand with the green cup because of their movement. As a result, the saliency map, which equally summed up

the five conspicuity maps, detected the three highly salient locations in the scene (see Figure 3 (a)). Note that our model selected the locations of which saliency was higher than the maximum $\times$ 0.9 in each frame, which allows us to evaluate the general tendency of parental actions. Through our experiment, the blue cup was not salient due to the blue background.

4.2 Results

4.2.1 Proportion of attended locations

We first compared how often a parent's face, his/her hands, and the cups were attended to by the model in IDI and in ADI. The attended locations were automatically classified using the predefined colors and positions of the targets. The results were compared separately in three time periods: before, during, and after the task. The start and the end of the task were defined when a parent picked up the first cup and when he/she put down the final cup into the blue one, respectively. The length of the periods before and after the task was 2 [sec].

Figures 4, **5**, and **6** show the results for the periods before, during, and after the task. In each graph, the horizontal axis denotes the label of the subjects, and the vertical axis denotes the proportion at which (a) a parent's face, (b) his/her hands, and (c) the cups were attended to over the period. When an attended location was at none of them, e.g., at a parent's cloth and at the tray, it was counted as (d) the others. The means and the standard deviations are listed in **Table 1**.

Before task: The non-parametric test (the Wilcoxon test) revealed significant differences in the proportion of attention on the cups (Figure 4 (c); $Z = -2.045,\ p < 0.05$) and in that on the others ((d); $Z = -1.988,\ p < 0.05$). It indicates that the cups attracted more attention in IDI than in ADI, and that the others were less attended to in IDI than in ADI.

During task: The non-parametric test revealed a significant difference in the proportion of attention on a parent's face (Figure 5 (a); $Z = -2.556,\ p < 0.05$). It also showed a statistical trend in the proportion of attention on parent's hands ((b); $Z = -1.817,\ p = 0.069$). A parent's face attracted much more attention in IDI than in ADI while his/her hands attracted less attention in IDI than in ADI.

After task: The non-parametric test revealed a statistical trend in the proportion of attention on a parent's face (Figure 6 (a); $Z = -1.874,\ p = 0.061$). The parametric t-test showed a trend in the proportion of attention on the cups ((c); $t(14) = 1.846,\ p = 0.086$). These results suggest that a parent's face was attended to in ADI more than in IDI, and that the cups were attended to in IDI more than in ADI.

4.2.2 Contribution of static features to saliency of objects

We next analyzed how much the static visual features of the cups contributed to their saliency in IDI and in ADI. Here the static features include the color, the intensity, and the orientation while the motion features include the flicker and the motion. The sum of the degrees of saliency derived from the static features was compared between IDI and ADI.

Figure 7 shows the contribution rate of the static features to the saliency of the cups (a) before, (b) during, and (c) after the task. **Table 2** lists the means and the standard deviations. The non-parametric test (the Wilcoxon test) revealed significant differences in the contribution rates before the task (Figure 7 (a); $Z = -2.040,\ p < 0.05$) and during the task ((b); $Z = -3.045,\ p < 0.05$). It indicates that in the two time periods the static features much more contributed to the saliency of the cups in IDI than in ADI.

Table 1. Proportions of attended locations

		IDI		ADI	
		M	SD	M	SD
before task	parent's face	0.070	0.104	0.049	0.047
	parent's hands	0.583	0.171	0.521	0.192
	cups	0.289	0.145	0.196	0.185
	others	0.216	0.184	0.356	0.220
during task	parent's face	0.040	0.038	0.019	0.017
	parent's hands	0.680	0.150	0.715	0.127
	cups	0.448	0.117	0.433	0.112
	others	0.089	0.088	0.089	0.083
after task	parent's face	0.085	0.103	0.154	0.117
	parent's hands	0.484	0.311	0.475	0.239
	cups	0.306	0.198	0.180	0.123
	others	0.230	0.232	0.270	0.176

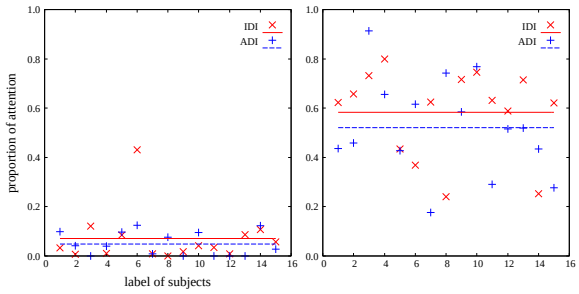
Table 2. Contribution of static features to saliency of cups

	IDI		ADI	
	M	SD	M	SD
before task	0.461	0.331	0.240	0.267
during task	0.256	0.203	0.090	0.100
after task	0.650	0.349	0.421	0.405

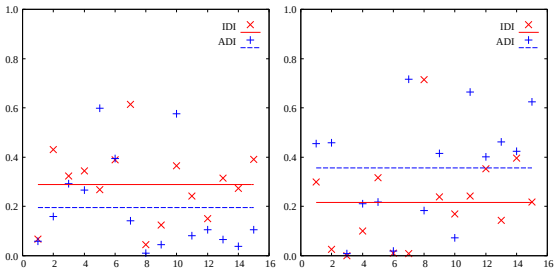
5 DISCUSSIONS

Our first focus of analysis revealed that a parent's face attracted much more attention in IDI than in ADI during the task while it attracted less attention in IDI than in ADI after the task. A reason is that the parents in IDI often talked to and smiled at their infants when demonstrating the task. They commented on each action while executing it, tried to maintain the infants' attention by addressing them verbally, and tried to get the infants interested in the task by showing emotional expressions. These behaviors caused movements on the parents' faces and made them more salient than others (see **Figure 8** (a)). By contrast, in ADI the parents rarely talked to or smiled at the adult partner during the task but explained the task after finishing it. Thus, their faces attracted more attention after the task. The result that the parents' hands were more attended to in ADI than in IDI during the task also indicates that their faces did not often move compared to their hands. We suggest from these results that parents give their infants immediate feedback on their actions, which helps infants to detect what actions are important and relevant to the task.

Our further analysis focusing on the objects involved in the task revealed that the objects were more salient in IDI than in ADI before and after the task. The saliency emerged because the parents interacting with their infants tended to put longer pauses before and after the task. While many of the parents in ADI started the task without checking whether the adult partner looked at the task-relevant locations, in IDI, they looked at the infants first and then started the task after confirming the infants' attention on the cups (see Figure 8 (b)). They also tried to attract the infants' attention on the cups by shaking them before the task. The result that the other locations attracted less attention in IDI than in ADI before the task also indicates that the parents made much effort to attract the attention of infants on the

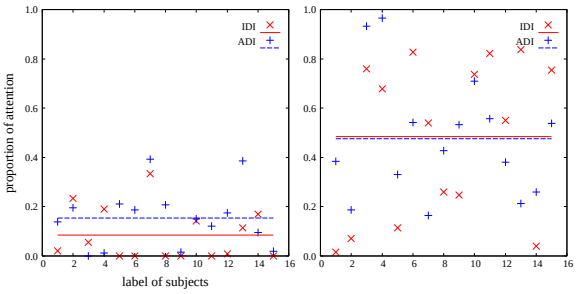


(a) parent's face (b) parent's hands

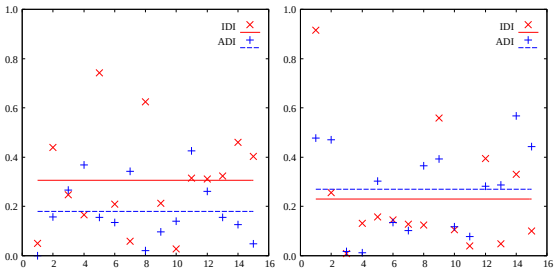


(c) cups (d) others

Figure 4. Proportions of attended locations before task (2 [sec])

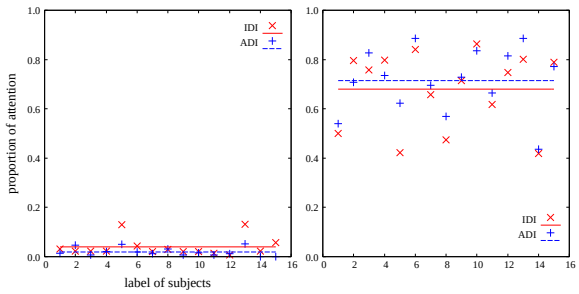


(a) parent's face (b) parent's hands

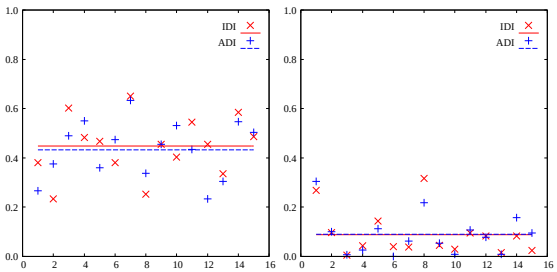


(c) cups (d) others

Figure 6. Proportions of attended locations after task (2 [sec])

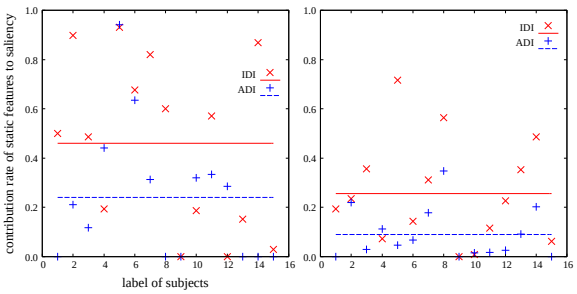


(a) parent's face (b) parent's hands

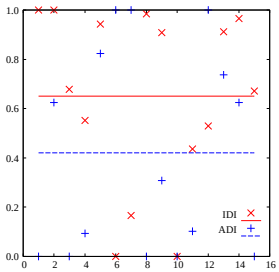


(c) cups (d) others

Figure 5. Proportions of attended locations during task



(a) before task (b) during task



(c) after task

Figure 7. Contribution of static features to saliency of cups

(a) parent's face attended to during task in IDI

(b) cups attended to before task in IDI

(c) cups attended to after task in IDI

Figure 8. Examples of attended locations, which are indicated by a red, a green, or a blue box if they are on a parent's face, on his/her hands, or on the cups, respectively

task-related locations. In addition, the parents in IDI tended to stop their movement and look at the infants for a while after the task (see Figure 8 (c)) while the parents in ADI continued to move and commented a lot on the task. They likely showed the goal state of the task to the infants. We therefore suggest that parents aid their infants detecting the initial and goal states of the actions by inserting longer pauses before and after the task.

Our analysis on the contribution of the static features to the saliency of the objects showed that the features of the color, the intensity, and the orientation of the cups contributed much more to their saliency in IDI than in ADI. When the cups are attended to as salient locations, two reasons are considered: motion and static visual features. In IDI the saliency of the cups was derived not only from their movement but also from their intrinsic features, i.e., the color, the intensity, and the orientation, while in ADI the saliency was mostly came from their movement. The reason is that the parents in IDI often stopped their movement during the demonstration of the task and tried to attract the infants' attention not on their hands' motion but on the cups they were holding. Thus, the cups were attended to as salient locations because of their intrinsic features. We suggest with these results that parental actions help infants to detect the static features of the objects, which consequently enables them to better perceive the physical structure of the objects.

Although these findings are already very significant, some results are considered to be improved. Our analysis, for example, found a trend but did not reveal a statistically significant difference between the proportions of attention on the cups in IDI and in ADI after the task. Before the experiment, we hypothesized that the cups would attract much more attention in IDI than in ADI after the task as before the task. The reason why the cups were not so salient after the task is the blue background. In the goal state, all of the green, the yellow, and the red cups were put in the blue one, which means only the blue one was visible. Thus, the blue cup in the blue background was not detected as a salient location. We will therefore analyze other tasks using other colored objects to evaluate our hypothesis.

The position of the camera which recorded parents' actions also can be optimized. The camera was set higher than the head position of infants so that the view of the camera was not occluded by the infants. This position caused less saliency of the parents' faces because they always looked down to gaze at infants. We will thus change the position of the camera so that we can analyze motionese from a real infant viewpoint.

6 CONCLUSION

Our analysis on parental actions using a saliency-based attention model revealed that motionese can help infants (1) to receive immediate social feedback on the actions, (2) to detect the initial and goal states of the objects used in the actions, and (3) to look at the static features of the objects. In imitation learning, immediate feedback on the actions may allow infants to detect what actions are important and should be imitated. To look at the initial and goal states of the objects may be helpful in understanding the intention of the actions and in imitating the actions not only at the trajectory level but also at the goal level. To attend to the static features of the objects may also help infants to perceive the structure and the configuration of the objects. Therefore, all these results indicate that parental actions contribute to highlight the meaningful structures of the actions. We conclude that motionese can help infants to detect "what to imitate" and that the saliency-based attention model enables a robot to leverage these advantages in its imitation learning.

In contrast to current studies on robot imitation, in which a robot was given the knowledge about task-related actions and/or the goal of actions, our analysis showed that motionese enables a robot to detect these features autonomously. The model of saliency-based visual attention could highlight them in the sequences of parental actions. However, to solve the problem of "what to imitate," we still need to answer the following question. Which characteristics of actions, i.e., the trajectory or the goal of actions, should be imitated? We intend to further analyze motionese with respect to this problem.

We will also address the issue of "how to imitate." A robot that attempts to imitate human actions has to know how to transform the human movement into its own movement. To approach this problem, we propose a simple mapping from human movement detected in a robot's vision to the motion primitives of the robot represented in its somatic sense is enough to make the robot roughly imitate the actions [15, 27]. The motion primitives are designed with a set of neurons that are responsible to different motion directions while human movement is also detected and represented with neurons that are responsible to different motion directions [27]. We will develop such a mechanism and evaluate together with the attention model if they enable robots to imitate human actions by leveraging motionese.

REFERENCES

[1] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn, 'Action, state and effect metrics for robot imitation', in *Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication*, (2006).

[2] Minoru Asada, Karl F. MacDorman, Hiroshi Ishiguro, and Yasuo Kuniyoshi, 'Cognitive developmental robotics as a new paradigm for the design of humanoid robots', *Robotics and Autonomous Systems*, **37**, 185–193, (2001).

[3] Aude Billard, 'Learning motor skills by imitation: A biologically inspired robotic model', *Cybernetics and Systems: An International Jounal*, **32**, 155–193, (2001).

[4] Aude G. Billard, Sylvain Calinon, and Florent Guenter, 'Discriminative and adaptive imitation in uni-manual and bi-manual tasks', *Robotics and Autonomous Systems*, **54**(5), 370–384, (2006).

[5] Rebecca J. Brand, Dare A. Baldwin, and Leslie A. Ashburn, 'Evidence for 'motionese': modifications in mothers' infant-directed action', *Developmental Science*, **5**(1), 72–83, (2002).

[6] Cynthia Breazeal and Brian Scassellati, 'A context-dependent attention system for a social robot', in *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 1146–1151, (1999).

[7] Cynthia Breazeal and Brian Scassellati, 'Challenges in building robots that imitate people', in *Imitation in Animals and Artifacts*, eds., K. Dautenhahn and C. L. Nehaniv, 363–389, MIT Press, (2002).

[8] Cynthia Breazeal and Brian Scassellati, 'Robots that imitate humans', *Trends in Cognitive Sciences*, **6**(11), 481–487, (2002).

[9] J. Gavin Bremner, *Infancy*, Blackwell Publishers Limited, 1994.

[10] Sylvain Calinon, Florent Guenter, and Aude Billard, 'Goal-directed imitation in a humanoid robot', in *Proceedings of the International Conference on Robotics and Automation*, (2005).

[11] Yiannis Demiris and Gillian Hayes, 'Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model', in *Imitation in Animals and Artifacts*, eds., K. Dautenhahn and C. L. Nehaniv, 321–361, MIT Press, (2002).

[12] Jannik Fritsch, Nils Hofemann, and Katharina Rohlfing, 'Detecting 'when to imitate' in a social context with a human caregiver', in *Proceedings of the ICRA Workshop on Social Mechanisms of Robot Programming by Demonstration*, (2005).

[13] Lakshmi J. Gogate and Lorraine E. Bahrick, 'Intersensory redundancy and 7-month-old infants' memory for arbitrary syllable-object relations', *Infancy*, **2**(2), 219–231, (2001).

[14] Lakshmi J. Gogate, Lorraine E. Bahrick, and Jilayne D. Watson, 'A study of multimodal motherese: The role of temporal synchrony between verbal labels and gestures', *Child Development*, **71**(4), 878–894, (2000).

[15] Verena V. Hafner and Yukie Nagai, 'Imitation behaviour evaluation in human robot interaction', in *Proceedings of the 6th International Workshop on Epigenetic Robotics*.

[16] L. Itti, N. Dhavale, and F. Pighin, 'Realistic avatar eye and head animation using a neurobiological model of visual attention', in *Proceedings of the SPIE 48th Annual International Symposium on Optical Science and Technology*, pp. 64–78, (2003).

[17] Laurent Itti, Christof Koch, and Ernst Niebur, 'A model of saliency-based visual attention for rapid scene analysis', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(11), 1254–1259, (1998).

[18] Jana M. Iverson, Olga Capirci, Emiddia Longobardi, and M. Cristina Caselli, 'Gesturing in mother-child interactions', *Cognitive Development*, **14**, 57–75, (1999).

[19] Joseph L. Jacobson, David C. Boersma, Robert B. Fields, and Karen L. Olson, 'Paralinguistic features of adult speech to infants and small children', *Child Development*, **54**, 436–442, (1983).

[20] I. Kiraly, B. Jovanovic, W. Prinz, G. Aschersleben, and G Gergely, 'The early origins of goal attribution in infancy', *Consciousness and Cognition*, **12**, 752–769, (2003).

[21] Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue, 'Learning by watching: Extracting reusable task knowledge from visual observation of human performance', *IEEE Transactions on Robotics and Automation*, **10**, 799–822, (1994).

[22] Nobuo Masataka, 'Motherese in a signed language', *Infant Behavior and Development*, **15**, 453–460, (1992).

[23] Nobuo Masataka, 'Perception of motherese in a signed language by 6-month-old deaf infants', *Developmental Psychology*, **32**(5), 874–879, (1996).

[24] Nobuo Masataka, 'Perception of motherese in japanese sign language by 6-month-old hearing infants', *Developmental Psychology*, **34**(2), 241–246, (1998).

[25] Andrew N. Meltzoff and M. Keith Moore, 'Imitation of facial and manual gestures by human neonates', *Science*, **198**, 75–78, (1977).

[26] Andrew N. Meltzoff and M. Keith Moore, 'Imitation in newborn infants: Exploring the range of gestures imitated and the underlying mechanisms', *Developmental Psychology*, **25**(6), 954–962, (1989).

[27] Yukie Nagai, 'Joint attention development in infant-like robot based on head movement imitation', in *Proceedings of the Third International Symposium on Imitation in Animals and Artifacts*, pp. 87–96, (2005).

[28] Chrystopher L. Nehaniv and Kerstin Dautenhahn, 'Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications', in *Interdisciplinary Approaches to Robot Learning, World Scientific Series in Robotics and Intelligent Systems*, eds., J. Demiris and A. Birk, volume 24, (2000).

[29] Chrystopher L. Nehaniv and Kerstin Dautenhahn, 'Like me? measures of correspondence and imitation', *Cybernetics and Systems: An International Journal*, **32**, 11–51, (2001).

[30] Katharina J. Rohlfing, Jannik Fritsch, Britta Wrede, and Tanja Jungmann, 'How can multimodal cues from child-directed interaction reduce learning complexity in robot?', *Advanced Robotics*, **20**(10), 1183–1199, (2006).

[31] Stefan Schaal, 'Is imitation learning the route to humanoid robots?', *Trends in Cognitive Science*, **3**, 233–242, (1999).

[32] Joachim Schmidt, Jannik Fritsch, and Bogdan Kwolek, 'Kernel particle filter for real-time 3d body tracking in monocular color images', in *Proceedings of the Automatic Face and Gesture Recognition*, pp. 567–572, (2006).

[33] J. A. Sommerville and A. L. Woodward, 'Pulling out the intentional structure of action: the relation between action processing and action production in infancy', *Cognition*, **95**, 1–30, (2005).

[34] Ales Ude, Curtis Man, Marcia Riley, and Christopher G. Atkeson, 'Automatic generation of kinematic models for the conversion of human motion capture data into humanoid robot motion', in *Proceedings of the First IEEE-RAS International Conference on Humanoid Robots*, (2000).

A Theoretical Consideration on Robotic Imitation of Human Action According to Demonstration plus Suggestion

Masao Yokota [1]

Abstract. The Mental Image Directed Semantic Theory (MIDST) has proposed an omnisensory mental image model and its description language $\boldsymbol{L_{md}}$ intended to facilitate intuitive human-system interaction such that happens between non-expert people and home robots. The most remarkable feature of $\boldsymbol{L_{md}}$ is its capability of formalizing both temporal and spatial event concepts on the level of human/robotic sensations. This paper presents a brief sketch of $\boldsymbol{L_{md}}$ and a theoretical consideration on robotic imitation of human action driven by human suggestion interpreted in $\boldsymbol{L_{md}}$, controlling the robotic attention mechanism efficiently.

1 INTRODUCTION

Robotic or artificial imitation is one kind of machine learning on human actions and there have been reported a considerable number of studies on imitation learning from human actions demonstrated without any verbal hint [e.g., 1-3]. In this case, it is extremely difficult for a robot to understand which part of human demonstration is significant or not because there are too many things to attend to as it is. That is, it is an important issue where the attention of the observer should be focused on when a demonstrator performs an action. Whereas there have been several proposals to control attention mechanisms efficiently in such top-down ways as guided by the prediction or strategy based on sensory data and knowledge of goals or tasks [e.g., 4, 5, 14], they are not realistic when a large number of actions must be imitated distinctively with various speeds, directions, trajectories, etc.

The author has been working on integrated multimedia understanding for intuitive human-robot interaction, that is, interaction between non-expert or ordinary people and home robots, where natural language is the leading information medium for their intuitive communication [6, 12]. For ordinary people, natural language is the most important because it can convey the exact intention of the sender to the receiver due to its syntax and semantics common to its users, which is not necessarily the case for another medium such as gesture or so. Therefore, the author believes that it is most desirable to realize robotic imitation aided by human verbal suggestion where robotic attention to human demonstration is efficiently controllable based on semantic understanding of the suggestion.

For such a purpose, it is essential to develop a systematically computable knowledge representation language (KRL) as well as representation-free technologies such as neural networks for processing unstructured sensory/motory data. This type of language is indispensable to *knowledge-based* processing such as *understanding* sensory events, *planning* appropriate actions and *knowledgeable* communication with ordinary people in natural language, and therefore it needs to have at least a good capability of representing spatiotemporal events that correspond to humans'/robots' sensations and actions in the real world.

Most of conventional methods have provided robotic systems with such quasi-natural language expressions as 'move(*Velocity*, *Distance*, *Direction*)', 'find(*Object*, *Shape*, *Color*)' and so on for human instruction or suggestion, uniquely related to computer programs to deploy sensors/ motors [e.g., 7, 8]. In association with robotic imitation intended here, however, these expression schemas are too linguistic or coarse to represent and compute sensory/motory events in an integrated way.

The Mental Image Directed Semantic Theory (MIDST) [9] has proposed a model of human attention-guided perception yielding omnisensory images that inevitably reflect certain movements of the focus of attention of the observer (FAO) scanning certain matters in the world. More analytically, these omnisensory images are associated with spatiotemporal changes (or constancies) in certain attributes of the matters scanned by FAO and modeled as temporally parameterized "loci in attribute spaces", so called, to be formulated in a formal language $\boldsymbol{L_{md}}$. This language has already been implemented on several types of computerized intelligent systems [e.g., 10, 12].

This paper presents a brief sketch of the formal language $\boldsymbol{L_{md}}$ and a theoretical consideration on robotic imitation of human demonstrated action aided by human suggestion interpreted as semantic expression in $\boldsymbol{L_{md}}$. The most remarkable feature of $\boldsymbol{L_{md}}$ is its capability of formalizing spatiotemporal matter concepts grounded in human/robotic sensation while the other similar KRLs are designed to describe the logical relations among conceptual primitives represented by lexical tokens [e.g., 11]. In $\boldsymbol{L_{md}}$ expression are hinted what and how should be attended to in human action as analogy of human FAO movement and thereby the robotic attention can be controlled in a top-down way.

2 A BRIEF SKETCH OF L_{md}

An attribute space corresponds with a certain measuring instrument just like a barometer, thermometer or so and the loci represent the movements of its indicator. For example, the moving black triangular object shown in Figure 1 is assumed to be perceived as the loci in the three attribute spaces, namely, those of 'Location', 'Color' and 'Shape' in the observer's brain.

[1] Fukuoka Institute of Technology, Japan, email: yokota@fit.ac.jp

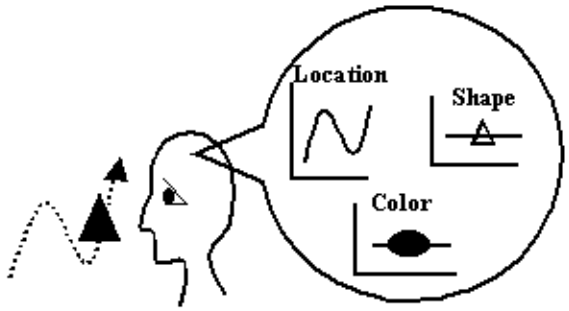


Figure1. Mental image model

Such a locus is to be articulated by "Atomic Locus" with an ***absolute*** time-interval $[t_i, t_f]$ $(t_i < t_f)$ as depicted in Figure 2 (up) and formulated as (1).

$$L(x,y,p,q,a,g,k) \quad (1)$$

This formula is called 'Atomic Locus Formula' whose first two arguments are often referred to as 'Event Causer (EC)' and 'Attribute Carrier (AC)', respectively. A logical combination of atomic locus formulas defined as a well-formed formula (i.e., wff) in predicate logic is called simply 'Locus Formula'. The intuitive interpretation of (1) is given as follows, where 'matter' refers to 'object' or 'event' largely.

"Matter 'x' causes Attribute 'a' of Matter 'y' to keep (p=q) or change (p ≠ q) its values temporally (g=G_t) or spatially (g=G_s) over a time-interval, where the values 'p' and 'q' are relative to the standard 'k'."

When $g=G_t$ and $g=G_s$, the locus indicates monotonic change or constancy of the attribute in time domain and that in space domain, respectively. The former is called 'temporal event' and the latter, 'spatial event'. For example, the motion of the 'bus' represented by S1 is a temporal event and the ranging or extension of the 'road' by S2 is a spatial event whose meanings or concepts are formulated as (2) and (3), respectively, where A_{12} denotes 'Physical Location'. These two formulas are different only at 'Event Type (i.e., *g*)'.

(S1) The bus runs from Tokyo to Osaka.

$$(\exists x,y,k)L(x,y,Tokyo,Osaka,A_{12},\mathbf{G_t},k)\wedge bus(y) \quad (2)$$

(S2) The road runs from Tokyo to Osaka.

$$(\exists x,y,k)L(x,y,Tokyo,Osaka,A_{12},\mathbf{G_s},k)\wedge road(y) \quad (3)$$

The author has hypothesized that the difference between temporal and spatial event concepts can be attributed to the relationship between the Attribute Carrier (AC) and the Focus of the Attention of the Observer (FAO) [9]. To be brief, it is assumed that the FAO is fixed on the whole AC in a temporal event but *runs* about on the AC in a spatial event. According to this assumption, as shown in Figure 3, the *bus* and the FAO move together in the case of S1 while the FAO solely moves along the *road* in the case of S2.

Any locus in a certain Attribute Space can be formalized as a combination of atomic locus formulas and, so called, tempo-logical connectives, among which the most frequently used are 'Simultaneous AND (Π)' and 'Consecutive AND (•)' as appear in the conceptual definition (4) of the English verb 'fetch' depicted in Figure 2 (down).

$$(\lambda x,y)fetch(x,y)\leftrightarrow(\lambda x,y)(\exists p_1,p_2,k)L(x,x,p_1,p_2,A_{12},G_t,k)\bullet ((L(x,x,p_2,p_1,A_{12},G_t,k)\Pi L(x,y,p_2,p_1,A_{12},G_t,k))\wedge x\neq y\wedge p_1\neq p_2 \quad (4)$$

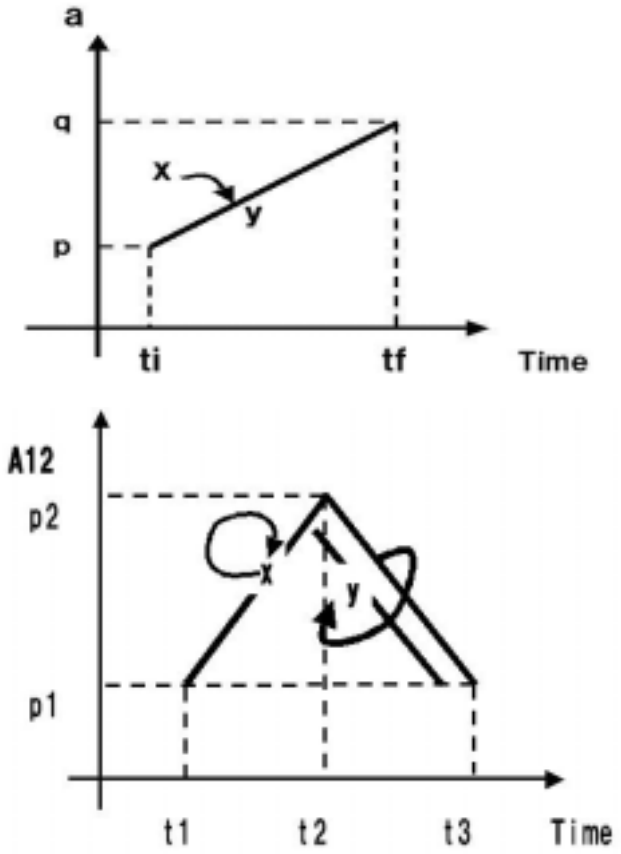


Figure 2. Atomic Locus (up) and Locus of 'fetch' (down)

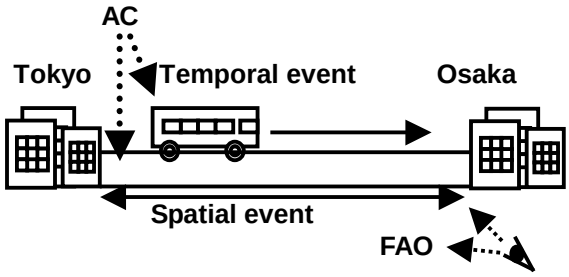


Figure 3. FAO movements and Event types

In order for explicit indication of time duration, 'Empty Event (EE)' denoted by 'ε' is introduced by the definition (5) with the attribute 'Time Point (A34)'. According to this scheme, the duration $[t_a, t_b]$ of an arbitrary locus χ can be expressed as (6).

$$\varepsilon([t_1,t_2])\leftrightarrow(\exists x,y,g,k)\ L(x,y,t_1,t_2,A34,g,k) \quad (5)$$

$$\chi\ \Pi\ \varepsilon([t_a, t_b]) \quad (6)$$

All the same way, an object concept is also defined and expressed in $\boldsymbol{L_{md}}$ as a combination of potential events on its properties and its relations with others. For example, the conceptual descriptions of 'rain', 'wind' and 'air' can be given as (7)-(9), reading 'Rain is water attracted from the sky by the earth, makes an object wetter, is pushed an umbrella to by a human,…,' 'Wind is air, affects the direction of rain,… ,' and 'Air has no shape, no taste, no vitality, …,' respectively.

$(\lambda x)\mathrm{rain}(x)\leftrightarrow(\lambda x)(\exists x_1,x_2,\ldots)L(\_,x,x_1,x_1,A_{41},G_t,\_)$
$\prod L(\mathrm{Earth},x,\mathrm{Sky},\mathrm{Earth},A_{12},G_t,\_)\prod L(x,x_2,p,q,A_{25},G_t,\_)$
$\prod L(x_3,x_4,x,x,A_{19},G_t,x_3)\wedge\mathrm{water}(x_1)$
$\wedge\mathrm{object}(x_2)\wedge\mathrm{human}(x_3)\wedge\mathrm{umbrella}(x_4)\wedge(p<q)\ldots$ (7)

$(\lambda x)\mathrm{wind}(x)\leftrightarrow(\lambda x)(\exists x_1,x_2,\ldots)L(\_,x,x_1,x_1,A_{41},G_t,\_)$
$\wedge\mathrm{air}(x_1)\wedge(L(x,x_2,p,q,A_{13},G_t,\_)\wedge\mathrm{rain}(x_2)\ldots$ (8)

$(\lambda x)\mathrm{air}(x)\leftrightarrow(\lambda x)(\ldots\wedge L^*(\_,x,/,/,A_{11},G_t,\_)\wedge\ldots\wedge$
$L^*(\_,x,/,/,A_{29},G_t,\_)\wedge\ldots\wedge L^*(\_,x,/,/,A_{39},G_t,\_)\wedge\ \ldots)$ (9)

Hereafter, for simplicity of $\boldsymbol{L_{md}}$ expression, the special symbols '*', '_'and '/' are often employed to represent 'always', 'something (or some value)' and 'nothing (no value)' as defined by (10)-(12), respectively.

$X^* \leftrightarrow (\forall[p,q])X\ \Pi\ \varepsilon([p,q])$ (10)

$L(\ldots,\_,\ldots)\leftrightarrow(\exists\omega)L(\ldots,\omega,\ldots)$ (11)

$L(\ldots,/,\ldots)\leftrightarrow\sim(\exists p)\ L(\ldots,\omega,\ldots)$ (12)

Table 1 shows about 50 attributes extracted exclusively from English and Japanese words of common use contained in certain thesauri [9]. Most of them (i.e., A01-A45) correspond to the sensory receptive fields in human brains. For example, those marked with '*' in this table can be associated to the sense 'sight'. Correspondingly, six categories of standards shown in Table 2 have been extracted that are necessary for representing relative values of each attribute in Table 1. ***These tables show that ordinary people live their casual lives, attending to tens of attributes of the matters in the world to cognize them in comparison with several kinds of standards.***

Table 1. List of attributes

Code	Attribute [Property†] (*words/phrases concerned*)
*A01	PLACE OF EXISTE NCE [N] (*happen, perish*)
*A02	LENGTH [S] (*long, shorten, close, away*)
*A03	HEIGHT [S] (*high, lower*)
*A04	WIDTH [S] (*widen, narrow*)
*A05	THICKNESS [S] (*thick, thin*)
*A06	DEPTH1 [S] (*deep, shallow*)
*A07	DEPTH2 [S] (*deep, concave*)
*A08	DIAMETER [S] (*across, in diameter*)
*A09	AREA [S] (*square meters, acre*)
*A10	VOLUME [S] (*litter, gallon*)
*A11	SHAPE [N] (*round, triangle*)
*A12	PHYSICAL LOCATION [N] (*move, stay*)
*A13	DIRECTION [N] (*turn, wind, left*)
*A14	ORIENTATION [N] (*orientate, command*)
*A15	TRAJECTORY [N] (*zigzag, circle*)
*A16	VELOCITY [S] (*fast, slow*)
*A17	MILEAGE [S] (*far, near*)
A18	STRENGTH OF EFFECT [S] (*strong, powerful*)
A19	DIRECTION OF EFFECT [N] (*pull, push*)
A20	DENSITY [S] (*dense, thin*)
A21	HARDNESS [S] (*hard, soft*)
A22	ELASTICITY [S] (*elastic, flexible*)
A23	TOUGHNESS [S] (*fragile, stiff*)
A24	TACTILE FEELING [S] (*rough, smooth*)
A25	HUMIDITY [S] (*wet, dry*)
A26	VISCOSITY [S] (*oily, watery*)
A27	WEIGHT [S] (*heavy, light*)
A28	TEMPERATURE [S] (*hot, cold*)
A29	TASTE [N] (*sour, sweet, bitter*)
A30	ODOUR [N] (*pungent, sweet*)
A31	SOUND [N] (*noisy, silent, loud*)
*A32	COLOR [N] (*red, white*)
A33	INTERNAL SENSATION [N] (*tired, hungry*)
A34	TIME POINT [S] (*o'clock, elapse*)
A35	DURATION [S] (*hour, minute, long, short*)
A36	NUMBER [S] (*ten, quantity, number*)
A37	ORDER [S] (*first, last*)
A38	FREQUENCY [S] (*sometimes, frequent*)
A39	VITALITY [S] (*alive, dead, vivid*)
A40	SEX [S] (*male, female*)
A41	QUALITY [N] (*make, destroy*)
A42	NAME [V] (*name, token*)
A43	CONCEPTUAL CATEGORY [V] (*mammal*)
*A44	TOPOLOGY [V] (*in, out, touch*)
*A45	ANGULARITY [S] (*sharp, dull, right angle*)
B01	WORTH [N] (*improve, praise, deny, alright*)
B02	LOCATION OF INFORMATION [N] (*tell, hear*)
B03	EMOTION [N] (*like, hate*)
B04	BELIEF VALUE [S] (*believe, trust*)
	……..

†S: scalar value, N: non-scalar value. *Attributes concerning the sense of sight.

Table 2. List of standards

Categories	Remarks
Rigid Standard	Objective standards such as denoted by measuring *units* (meter, gram, etc.).
Species Standard	The *attribute value ordinary* for a species. A *short train* is ordinarily longer than a *long pencil*.
Proportional Standard	'*Oblong*' means that the width is greater than the height at a physical object.
Individual Standard	*Much* money for one person can be too *little* for another.
Purposive Standard	One room large enough for a person's *sleeping* must be too small for his *jogging*.
Declarative Standard	The origin of an order such as 'next' must be declared explicitly just as 'next *to him*'.

3 INTELLIGENT SYSTEM IMAGES-M

3.1 System configuration

The intelligent system IMAGES-M [e.g., 10, 12] is assumed to be the main intelligence of the robot intended here. As shown in Figure 4, IMAGES-M is one kind of expert system equipped with five kinds of user interfaces for multimedia communication, that is, Sensory Data Processing Unit (SDPU), Speech Processing Unit (SPU), Picture Processing Unit (PPU), Text Processing Unit (TPU), and Action Data Processing Unit (ADPU) besides Inference Engine (IE) and Knowledge Base (KB). Each processing unit in collaboration with IE performs mutual conversion between each type of information medium and locus formulas.

IMAGES-M is a language-centered intelligent system in order to facilitate intuitive interaction between humans and robots. For comprehensible communication with humans, robots must understand natural language *semantically* and *pragmatically*. Here, as shown in Figure 5, semantic understanding means associating symbols to conceptual images of matters (i.e., objects or events), and pragmatic understanding means anchoring symbols to real matters by unifying conceptual images with perceptual images.

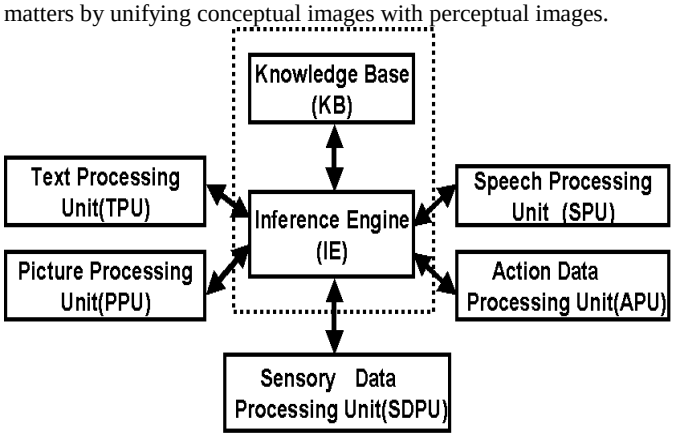


Figure 4. Configuration of IMAGES-M

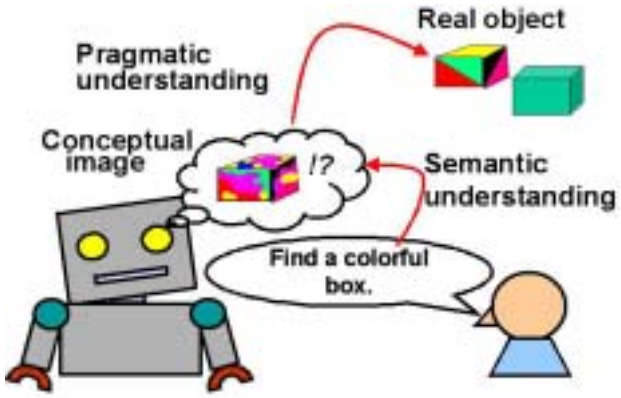


Figure 5. Semantic and pragmatic understanding

3.2 Semantic understanding

As shown in Figure 6, natural language expression (i.e, surface structure) and $\boldsymbol{L_{md}}$ expression (i.e., conceptual structure) are mutually translatable through surface dependency structure by utilizing syntactic rules and word meaning descriptions [9].

A word meaning description M_w is defined by (13) as a pair of 'Concept Part (C_p)' and 'Unification Part (U_p)'.

$$M_w \leftrightarrow [C_p : U_p] \quad (13)$$

The C_p of a word W is a locus formula about properties and relations of the matters involved such as shapes, colors, functions, potentialities, etc while its U_p is a set of operations for unifying the C_ps of W's syntactic governors or dependents. For example, the meaning of the English verb 'carry' can be given by (14).

$[(\exists x,y,p_1,p_2)\ L(x,x,p_1,p_2,A12,Gt,\_)\Pi$
$L(x,y,p_1,p_2,A12,Gt,\_)\wedge x\neq y\wedge p_1\neq p_2$:ARG(Dep.1,x);
ARG(Dep.2,y);] (14)

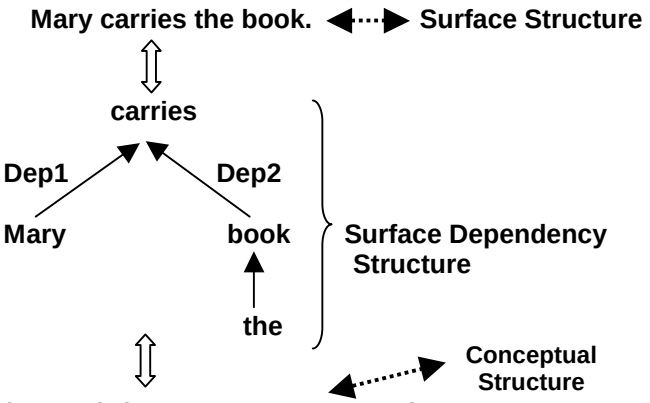


Figure 6. Mutual conversion between natural language and $\boldsymbol{L_{md}}$

(Input)
With the long red stick Tom precedes Jim.
(Output)
Tom with the long red stick goes before Jim goes.
Jim goes after Tom goes with the long red stick.
Jim follows Tom with the long red stick.
Tom carries the long red stick before Jim goes.
……

Figure 7. Paraphrasing as semantic understanding by IMAGES-M

The U_p above consists of two operations to unify the first dependent (Dep.1) and the second dependent (Dep.2) of the current word with the variables *x* and *y*, respectively. Here, Dep.1 and Dep.2 are the 'subject' and the 'object' of 'carry', respectively. Therefore, the surface structure '*Mary carries a book*' is translated into the conceptual structure (15) via the surface dependency structure shown in Figure 6.

$(\exists y,p_1,p_2)L(Mary,Mary,p_1,p_2,A12,Gt,\_)\Pi$
$L(Mary,y,p_1,p_2,A12,Gt,\_)\wedge Mary\neq y\wedge p_1\neq p_2\wedge book(y)$ (15)

For another example, the meaning description of the English preposition 'through' is also given by (16).

$[(\exists x,y,p_1,z,p_3,g,p_4)(\underline{L(x,y,p_1,z,A12,g,\_)}\bullet$
$L(x,y,z,p_3,A12,g,\_))\Pi\ L(x,y,p_4,p_4,A13,g,\_)\wedge p_1\neq z\wedge z\neq p_3$
:ARG(Dep.1,z); IF(Gov=Verb)→PAT(Gov,(1,1));
IF(Gov=Noun)→ARG(Gov,y);] (16)

The U_p above is for unifying the C_ps of the very word, its governor (Gov, a verb or a noun) and its dependent (Dep.1, a noun). The second argument (1,1) of the command PAT indicates the underlined part of (13) and in general (*i*,*j*) refers to the partial formula covering from the *i*th to the *j*th atomic formula of the current C_p. This part is the pattern common to both the C_ps to be unified. This is called 'Unification Handle (U_h)' and when missing, the C_ps are to be combined simply with '∧'.

Therefore the sentences S3, S4 and S5 are interpreted as (17)-(19), respectively. The underlined parts of these formulas are the results of PAT operations. The expression (20) is the C_p of the adjective 'long' implying 'there is some value greater than some standard of 'Length (A02)' which is often simplified as (20').

(S3) The train runs through the tunnel.
$(\exists x,y,p_1,z,p_3,p_4)(\underline{L(x,y,p_1,z,A12,Gt,\_)}\bullet$
$L(x,y,z,p_3,A12,Gt,\_))\Pi\ L(x,y,p_4,p_4,A13,Gt,\_)$
$\wedge p_1\neq z\ \wedge z\neq p_3 \wedge \text{train}(y)\ \wedge \text{tunnel}(z)$ (17)

(S4) The path runs through the forest.
$(\exists x,y,p_1,z,p_3,p_4)(\underline{L(x,y,p_1,z,A12,Gs,\_)}\bullet$
$L(x,y,z,p_3,A12,Gs,\_))\Pi\ L(x,y,p_4,p_4,A13,Gs,\_)$
$\wedge p_1\neq z\ \wedge z\neq p_3\ \wedge \text{path}(y)\ \wedge \text{forest}(z)$ (18)

(S5) The path through the forest is long.
$(\exists x,y,p_1,z,p_3,x_1,q,p_4,k_1)$
$(L(x,y,p_1,z,A12,Gs,\_)\bullet L(x,y,z,p_3,A12,Gs,\_))$
$\Pi\ L(x,y,p_4,p_4,A13,Gs,\_)\ \wedge L(x_1,y,q,q,A02,Gt,k_1)$
$\wedge p_1\neq z\wedge z\neq p_3\wedge q>k_1\wedge \text{path}(y)\wedge \text{forest}(z)$ (19)
$(\exists x_1,y_1,q,k_1)L(x_1,y_1,q,q,A02,Gt,k_1)\wedge q>k_1$ (20)
$(\exists x_1,y_1,k_1)L(x_1,y_1,\text{Long},\text{Long},A02,Gt,k_1)$ (20')

The process above is completely reversible except that multiple natural expressions as paraphrases can be generated by TPU in IMAGES-M as shown in Figure 7 because such event patterns as shown in Figure 2 are sharable among multiple word concepts. This is one of the most remarkable features of MIDST and is also possible between different languages as understanding-based translation [10, 12].

3.3 Pragmatic understanding

An event expressed in $\boldsymbol{L_{md}}$ is compared to a movie film recorded through a floating camera because it is necessarily grounded in FAO's movement over the event. For example, it is not the 'path' but the 'FAO' that 'sinks' in S6 or 'rises' in S7. Therefore, such expressions refer to the same scene pragmatically in spite of their appearances, whose semantic descriptions are given as (21) and (22), respectively, where 'A_{13}', '↑' and '↓' refer to the attribute 'Direction', and its values 'upward' and 'downward', respectively. This fact is generalized as '***Postulate of Reversibility of a Spatial Event*** (PRS)' belonging to people's intuitive knowledge about geography, and the conceptual descriptions (21) and (22) are called ***equivalent in the PRS***.

(S6) The path sinks to the brook.
$(\exists x,y,p,z)L(x,y,p,z,A_{12},G_s,\_)\Pi L(x,y,\downarrow,\downarrow,A_{13},G_s,\_)$
$\wedge \text{path}(y)\ \wedge \text{brook}(z)\ \wedge p\neq z$ (21)

(S7) The path rises from the brook.
$(\exists x,y,p,z)L(x,y,z,p,A_{12},G_s,\_)\Pi L(x,y,\uparrow,\uparrow,A_{13},G_s,k_2)$
$\wedge \text{path}(y)\ \wedge \text{brook}(z)\ \wedge p\neq z$ (22)

For another example of spatial event, Figure 8 (up) concerns human perception of the formation of multiple distinct objects, where FAO runs along an imaginary object so called 'Imaginary Space Region (ISR)'. This spatial event can be verbalized as S8 using the preposition 'between' and formulated as (22), corresponding also to such concepts as 'row', 'line-up', etc. Any type of topological relation between two objects is also to be formulated by employing an ISR. For example, S9 is translated into (23) or (23'), where '*In*', and '*Cont*' are the values 'inside' and 'contains' of the attribute 'Topology (A44)' represented by 3x3 matrices at the Sandard of '9-intersection model (*IM*)' [13], where '*In'* and '*Cont'* are the transposes each other.

(S8) □ is between Δ and ○.
$(\exists y,p)(L(\_,y,\Delta,\square,A_{12},G_s,\_)\bullet L(\_,y,\square,\circ,A_{12},G_s,\_))\Pi$
$L(\_,y,p,p,A_{13},G_s,\_)\ \wedge \text{ISR}(y)$ (22)

(S9) □ is in the room.
$(\exists x,y)L(\_,x,y,\square,A12,Gs,\_)\Pi L(\_,x,In,In,A44,Gt,IM)$
$\wedge \text{ISR}(x)\wedge \text{room}(y)$ (23)
$(\exists x,y)L(\_,x,\square,y,A12,Gs,\_)\Pi L(\_,x,Cont,Cont,A44,Gt,IM)$
$\wedge \text{ISR}(x)\wedge \text{room}(y)$ (23')

For more complicated examples, consider S10 and S11. The underlined parts are deemed to refer to some events neglected in time and in space, respectively. These events correspond with skipping of FAOs and are called 'Temporal Empty Event' and 'Spatial Empty Event', denoted by 'ε_t' and 'ε_s' as Empty Events with $g=G_t$ and $g=G_s$ at (5), respectively. Their concepts are described as (24) and (25), where 'A_{15}' and 'A_{17}' represent the attribute 'Trajectory' and 'Mileage', respectively. From the viewpoint of pragmatic understanding, the formula (25) can refer to such a spatial event depicted as the still picture in Figure 8 (down) while (24), a temporal event to be recorded as a movie.

(S10) The *bus* runs 10km straight east from A to B, and *<u>after a while</u>*, at C it meets the street with the sidewalk.
$(\exists x,y,z,p,q)(L(\_,x,A,B,A_{12},G_t,\_)\Pi$
$L(\_,x,0,10\text{km},A_{17},G_t,\_)\Pi L(\_,x,\text{Point},\text{Line},A_{15},G_t,\_)\Pi$
$L(\_,x,\text{East},\text{East},A_{13},G_t,\_))\bullet\varepsilon_t\bullet(L(\_,x,p,C,A_{12},G_t,\_)$
$\Pi L(\_,y,q,C,A_{12},G_s,\_)\Pi L(\_,z,y,y,A_{12},G_s,\_))$
$\wedge \text{bus}(x)\wedge \text{street}(y)\wedge \text{sidewalk}(z)\wedge p\neq q$ (24)

(S11) The *road* runs 10km straight east from A to B, and *<u>after a while</u>*, at C it meets the street with the sidewalk.
$(\exists x,y,z,p,q)(L(\_,x,A,B,A_{12},G_s,\_)\Pi$
$L(\_,x,0,10\text{km},A_{17},G_s,\_)\Pi L(\_,x,\text{Point},\text{Line},A_{15},G_s,\_)\Pi$
$L(\_,x,\text{East},\text{East},A_{13},G_s,\_))\bullet\varepsilon_s\bullet(L(\_,x,p,C,A_{12},G_s,\_)$
$\Pi L(\_,y,q,C,A_{12},G_s,\_)\Pi L(\_,z,y,y,A_{12},G_s,\_))$
$\wedge \text{road}(x)\wedge \text{street}(y)\wedge \text{sidewalk}(z)\wedge p\neq q$ (25)

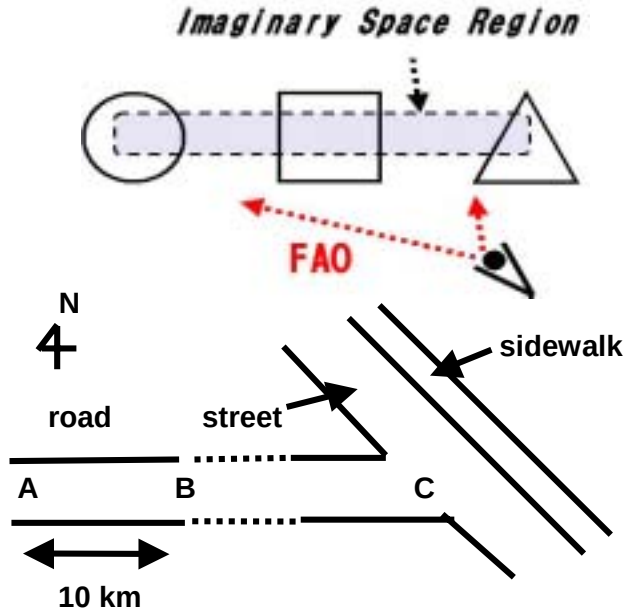


Figure 8. Complicated spatial events: 'row' (up) and 'example of road map' (down)

(a) A map generated from a locus formula by IMAGES-M

H: How does the national road run?
S: It extends between Pref. A and Pref. C via Pref. B.
H: Where does the bus go from the rail way station A?
S: It reaches the town D.
H: What is between the buildings A and B?
S: The railway D.
H: Where do the street A and the road B meet?
S: At the crossing C.
H: Where do the street A and the road B separate?
S: At the crossing C.

(b) Q-A on the map (a) by human (H) and IMAGES-M (S)

Figure 9. Cross-media operations as pragmatic understanding

Figures 9 (b) shows an example of question-answering on the real map (a) between a human and IMAGES-M [6, 10, 12], where the map is a pictorial interpretation of a locus formula by PPU. The system understood the query texts pragmatically by anchoring them to the map as a model of the real world, utilizing effectively several kinds of intuitive postulates such as PRS, as a matter of course, where distinction between temporal and spatial events is crucially important.

4 IMITATION GUIDED BY SUGGESTION

4.1 Definition

As shown in Figures 10 and 11, robotic imitation intended here is defined as a human-robot interaction where a human presents a robot a pair of demonstration and suggestion that is the expression of his/her intention and it behaviouralizes its conception, namely, the result of semantic and pragmatic understanding of the suggestion.

The processes shown in Figures 10 and 11 can be formalized as follows, where the pair of $\mathbf{P_i}$ and $\mathbf{Def_i}$ is called 'Conception' for the i-th imitation and denoted by $\mathbf{C_i}$.

$\mathbf{Int_i} \Rightarrow \mathbf{T_i}$, $\mathbf{D_i}$
$\mathbf{T_i}$, $\mathbf{K_L} \Rightarrow \mathbf{S_i}$
$\mathbf{D_i}$, $\mathbf{K_D} \Rightarrow \mathbf{Per_i}$
$\mathbf{S_i}$, $\mathbf{Per_i}$, $\mathbf{K_D} \Rightarrow \mathbf{P_i}$, $\mathbf{Def_i}$ (= $\mathbf{C_i}$)
$\mathbf{P_i}$, $\mathbf{Def_i}$, $\mathbf{K_D} \Rightarrow \mathbf{I_i}$

, where
$\mathbf{Int_i}$: The i-th intention by the human,
$\mathbf{T_i}$: The i-th suggestion by the human,
$\mathbf{S_i}$: Result of semantic understanding of the i-th suggestion,
$\mathbf{K_L}$: Linguistic knowledge in the robot,
$\mathbf{D_i}$: The i-th demonstration by the human,
$\mathbf{K_D}$: Domain-specific knowledge in the robot at the i-th session,
$\mathbf{Per_i}$: Perception of the i-th demonstration,
$\mathbf{P_i}$: Result of pragmatic understanding of the i-th suggestion,
$\mathbf{Def_i}$: Default specification for the i-th imitation,
$\mathbf{I_i}$: The i-th imitation by the robot,
$\Rightarrow$: Conversion process (e.g., inference, translation).

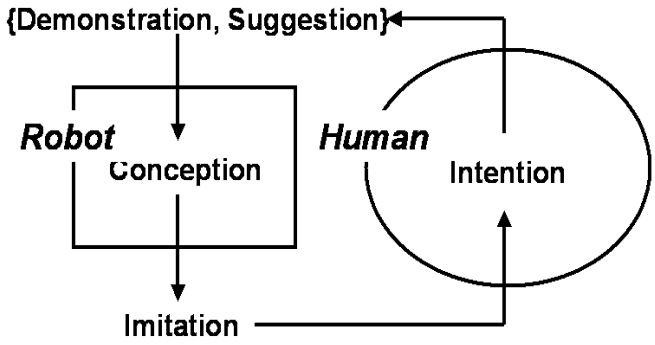


Figure 10. Imitation as human-robot interaction

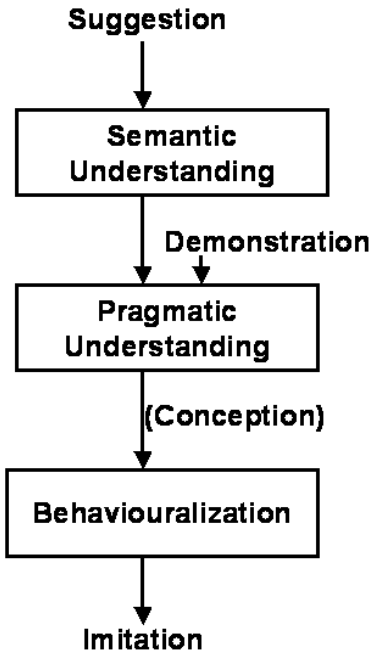


Figure 11. Imitation guided by suggestion

4.2 Theoretical simulation

As shown in Figure 10, it is assumed that there is a feedback loop between a human and a robot in order for the human to improve his/her previous suggestion or demonstration and for the robot to correct its previous imitation. For example, consider the scenario presented below and depicted in Figure12.

Scenario :

Robby is an intelligent humanoid robot and Tom is his user. Robby is called by Tom and enters Tom's room. This is Robby's first visit there. Robby sees Tom leftward and the brown pillar forward (, but doesn't see the green box or the yellow table). After a while, Tom tells Robby "Imitate me to my demonstration and suggestion."……

Here is described a theoretical simulation of the robotic imitation driven by the top-down control of the attention mechanism, which is almost that of problem finding/solving in the filed of AI [6, 12].

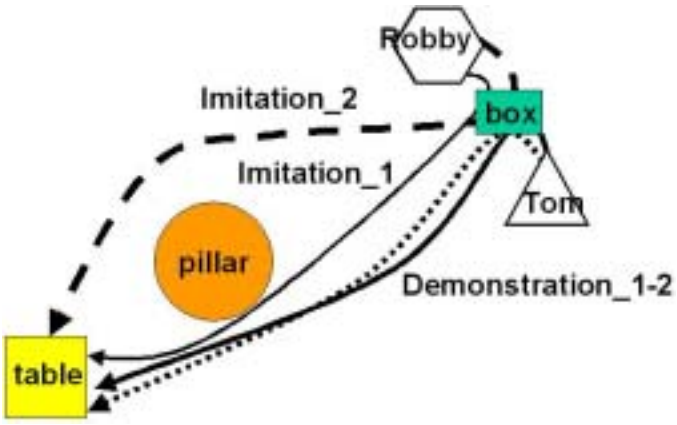


Figure 12. Tom's demonstrations and Robby's imitations

The sequence of the events assumed to happen is as follows.

[Robby's Perception of the initial situation, $\boldsymbol{Sit_0}$]

$\boldsymbol{Sit_0}\leftrightarrow$ L(_,O_{21},Brown,Brown,A_{32},G_t,_)Π
L(_,O_{22},Robby,Tom,A_{12},G_s,_)Π
L(_,O_{22},Lw_{21},Lw_{21},A_{13},G_s,Robby)Π
L(_,O_{23},Robby,O_{21},A_{12},G_s,_)Π
L(_,O_{23},Fw_{21},Fw_{21},A_{13},G_s,Robby)
∧pillar(O_{21})∧ISR(O_{22})∧ISR(O_{23})

Robby's perception of the situation (i.e., the underlined part of the scenario) is still rough due to its economical working mode that is to be specified by each Standard (or precision). The attributes A_{32} *and* A_{13} *are 'Color' and 'Direction', respectively. The values* Fw_{21} *and* Lw_{21} *stand for 'forward' and 'leftward' viewed from Robby as designated at the Standard, respectively.*

[Tom's Intention_1, $\boldsymbol{Int_1}$]

$\boldsymbol{Int_1}\leftrightarrow$L(Robby,Robby, O_{11},O_{13},A_{12},G_t,_)Π
L(Robby,O_{11},Robby,Robby,A_{12},G_t,_)Π
(L(_,O_{14},Tom,O_{11},A_{12},G_s,_)•L(_,O_{14},O_{11},Robby,A_{12},G_s,_))Π
L(_,O_{14},D_{11},D_{11},A_{13},G_s,_)ΠL(Robby,Robby,V_{11},V_{11},A_{16},G_t,_)ΠL
(_,O_{15},Robby,O_{12},A_{12},G_s,_)ΠL(Robby,O_{15},Dis,Dis,A_{44},G_t,_)
∧box(O_{11})∧pillar(O_{12})∧table(O_{13})∧ISR(O_{14})∧ISR(O_{15})

This formula implies that Tom wants Robby to carry the box between them to the table at a certain 'Velocity(A_{16})*',* V_{11} *without touching the pillar on the way, where '*O_{11}*' and '*O_{13}*' as the values of* A_{12} *represent their locations at each time point, and '*D_{11}*' is the direction to the box and Robby viewed from Tom.*

Tom is conscious that every attribute value to specify Robby's action is essentially vague but he believes that it should be imitated within certain tolerance associated with each Standard. The values **Dis** *and* **Meet** *stand for 'disjoint' and 'meet (or touch)' in Topology*(A_{44})*, respectively.*

<SESSION_1>

[Tom's Suggestion_1, $\boldsymbol{T_1}$ and Demonstration_1, $\boldsymbol{D_1}$]

$\boldsymbol{Int_1}\Rightarrow\boldsymbol{T_1}, \boldsymbol{D_1}$
$\boldsymbol{T_1}\leftrightarrow$ "Go to the table with the box between us like this."
$\boldsymbol{D_1}\leftrightarrow$ Figure 12

Tom decides to verbalize only the underlined part of Intention_1, $\boldsymbol{Int_1}$ *saliently with the belief that the rest can be included in his demonstration. Tom converts (or translates)* $\boldsymbol{Int_1}$ *into* $\boldsymbol{T_1}$ *and* $\boldsymbol{D_1}$*.*

[Robby's Semantic_Understanding_1, $\boldsymbol{S_1}$]

$\boldsymbol{T_1}, \boldsymbol{K_L}\Rightarrow\boldsymbol{S_1}$
$\boldsymbol{S_1}\leftrightarrow$(∃ x_1,x_2,x,y,z,p)L(x_2,x_2,y,x,A_{12},G_t,_)Π
L(x_2,y, x_2,x_2,A_{12},G_t,_)Π (L(_,z,x_2,y,A_{12},G_s,_)•
L(_,z,y,x_1,A_{12},G_s,_))ΠL(_,z,p,p,A_{13},G_s,_)
∧x_2≠x∧x_2≠y∧box(y)∧table(x)∧ISR(z)
∧person_1(x_1)∧person_2(x_2)

Robby interprets $\boldsymbol{T_1}$ *into* $\boldsymbol{S_1}$*. The variable '*x*' or '*y*' is not yet anchored to the 'real table' or the 'real box' in the real environment because Robby has not perceived them yet. The predicates '*person_1*' and '*person_2*' refer to the first person (I) and the second person (You) and are to be pragmatically understood as 'Tom' and 'Robby', respectively.*

[Robby's Pragmatic_Understanding_1, $\boldsymbol{P_1}$ and Default_1, $\boldsymbol{Def_1}$]

$\boldsymbol{D_1}\Rightarrow\boldsymbol{Per_1}$
$\boldsymbol{S_1}, \boldsymbol{Per_1}, \boldsymbol{K_D}\Rightarrow\boldsymbol{P_1}, \boldsymbol{Def_1}$
$\boldsymbol{P_1}\leftrightarrow$L(Robby,Robby,$O_{24}$,$O_{25}$,$A_{12}$,$G_t$,_)Π
L(Robby,O_{24},Robby,Robby,A_{12},G_t,_)Π
(L(_,O_{26},Robby,O_{25},A_{12},G_s,_)•L(_,O_{26},O_{25},Tom,A_{12},G_s,_))Π
L(_,O_{26},Lw_{21},Lw_{21},A_{13},G_s,_)∧box(O_{24})∧table(O_{25})∧ISR(O_{26})
$\boldsymbol{Def_1}\leftrightarrow$ L(Robby,Robby,1m/sec,1m/sec,A_{16},G_t,_)∧…

The 'Location (A_{12})*' is attended to according to* $\boldsymbol{S_1}$*.* $\boldsymbol{Per_1}$ *makes Robby aware that the words 'box' and 'table' should be anchored to the 'green object* O_{24}*' and the 'yellow object* O_{25}*' behind the pillar in the real environment, respectively. Robby conceives that he should approach to the table at his certain Standard.* $\boldsymbol{Def_1}$ *is inferred from* $\boldsymbol{Per_1}$ *and* $\boldsymbol{K_D}$ *as the default specification for the attributes not explicit in* $\boldsymbol{T_1}$*.*

[Robby's Imitation_1, $\boldsymbol{I_1}$]

$\boldsymbol{P_1}, \boldsymbol{Def_1}, \boldsymbol{K_D}\Rightarrow\boldsymbol{I_1}$
$\boldsymbol{I_1}\leftrightarrow$Figure12

Robby imitates $\boldsymbol{D_1}$ *according to* $\boldsymbol{P_1}$*,* $\boldsymbol{Def_1}$ *and* $\boldsymbol{K_D}$*.*

----- Resetting the situation to the initial situation $\boldsymbol{Sit_0}$-----

<SESSION_2>

[Tom's Suggestion_2, $\boldsymbol{T_2}$ and Demonstration_2, $\boldsymbol{D_2}$]

$\boldsymbol{I_1}\Rightarrow\boldsymbol{PI_1}$
$\boldsymbol{Int_1}, \boldsymbol{\sim PI_1}\Rightarrow\boldsymbol{Int_2}$
$\boldsymbol{Int_2}\Rightarrow\boldsymbol{T_2}, \boldsymbol{D_2}$
$\boldsymbol{T_2}\leftrightarrow$"Don't touch the pillar."
$\boldsymbol{D_2}\leftrightarrow$ Figure 12

Tom perceives $\boldsymbol{I_1}$ *as* $\boldsymbol{PI_1}$*. He denies* $\boldsymbol{PI_1}$ *and creates* $\boldsymbol{Int_2}$ *followed by* $\boldsymbol{T_2}$ and $\boldsymbol{D_2}$*.*

[Robby's Semantic_Understanding_2, $\boldsymbol{S_2}$]

$\boldsymbol{T_2}$, $\boldsymbol{K_L} \Rightarrow \boldsymbol{S_2}$

$\boldsymbol{S_2} \leftrightarrow (\exists x)L(\_,y,Robby,O_{21},A_{12},G_s,\_)$

$\Pi \sim L(Robby,x,Dis,Meet,A_{44},G_t,\_) \wedge ISR(x) \wedge pillar(O_{21})$

Robby gets aware that his imitation has been denied at the change of attribute 'Topology (A_{44})*' from '***Dis**joint*' to '***Meet***'.*

[Robby's Pragmatic_Understanding_2, $\boldsymbol{P_2}$ and Default_2, $\boldsymbol{Def_2}$]

$\boldsymbol{D_2} \Rightarrow \boldsymbol{Per_2}$

$\boldsymbol{S_2}$, $\boldsymbol{Per_2}$, $\boldsymbol{K_D} \Rightarrow \boldsymbol{P_2}$, $\boldsymbol{Def_2}$

$\boldsymbol{P_2} \leftrightarrow \boldsymbol{P_1} \wedge \underline{L(\_,O_{27},Robby,O_{21},A_{12},G_s,\_)\Pi}$

$\underline{L(Robby,O_{27},Dis,Dis,A_{44},G_t,\_)} \wedge pillar(O_{21}) \wedge ISR(O_{27})$

$\boldsymbol{Def_2} \leftrightarrow L(Robby,Robby,1m/sec,1m/sec,A_{16},G_t,\_) \wedge \ldots$

According to $\boldsymbol{S_2}$*, the 'Location* (A_{12})*' of Robby and the pillar and their 'Topology* (A_{44})*' are especially attended to, and the underlined part is conceived in addition to* $\boldsymbol{P_1}$*. No special attention is paid to the other attributes unmentioned yet.*

[Robby's Imitation_2, $\boldsymbol{I_2}$]

$\boldsymbol{P_2}$, $\boldsymbol{Def_2}$, $\boldsymbol{K_D} \Rightarrow \boldsymbol{I_2}$

$\boldsymbol{I_2} \leftrightarrow$ Figure 12

-----Resetting the situation to the initial situation $\boldsymbol{Sit_0}$-----

<SESSION_3>

[Tom's Suggestion_3, $\boldsymbol{T_3}$ and Demonstration_3, $\boldsymbol{D_3}$]

$\boldsymbol{I_2} \Rightarrow \boldsymbol{PI_2}$

$\boldsymbol{Int_2}$, $\boldsymbol{\sim PI_2} \Rightarrow \boldsymbol{Int_3}$ ($\leftrightarrow$***Null***)

$\boldsymbol{Int_3} \Rightarrow \boldsymbol{T_3}$, $\boldsymbol{D_3}$

$\boldsymbol{T_3} \leftrightarrow$"Alright."

$\boldsymbol{D_3} \leftrightarrow$ ***Null***

Tom fails to deny $\boldsymbol{PI_2}$ *and comes to have no other intention (*$\boldsymbol{Int_3} \leftrightarrow$***Null***). That is, Tom is satisfied by* $\boldsymbol{I_2}$ *and only tells Robby* "Alright."

[Robby's Semantic_Understanding_3, $\boldsymbol{S_3}$]

$\boldsymbol{T_3}$, $\boldsymbol{K_L} \Rightarrow \boldsymbol{S_3}$

$\boldsymbol{S_3} \leftrightarrow (\exists x,y,k)L(x,y,1,1,B_{01},G_t,k) \wedge person(x)$

Tom gets aware that something 'y' *has evaluated by some person* 'x' *as perfect* '1' *at 'Worth* (B_{01})*' with a certain Standard* 'k'.

[Robby's Pragmatic_Understanding_3, $\boldsymbol{P_3}$ and Default_3, $\boldsymbol{Def_3}$]

$\boldsymbol{S_3}$, $\boldsymbol{Per_3}$, $\boldsymbol{K_D} \Rightarrow \boldsymbol{P_3}$, $\boldsymbol{Def_3}$

$\boldsymbol{P_3} \leftrightarrow L(Tom,\boldsymbol{I_2},1,1,B_{01},G_t,Tom) \wedge person(Tom)$

$\boldsymbol{Def_3} \leftrightarrow L(Robby,\boldsymbol{I_3},/,/,A_{01},G_t,\_)$

Finally, Robby pragmatically conceives that Tom is satisfied by $\boldsymbol{I_2}$ *at Tom's Standard and believes that the next imitation,* $\boldsymbol{I_3}$ *is not needed to take 'Place of Existence* (A_{01})*'.*

[Robby's Imitation_3, $\boldsymbol{I_3}$]

$\boldsymbol{P_3}$, $\boldsymbol{Def_3}$, $\boldsymbol{K_D} \Rightarrow \boldsymbol{I_3}$

$\boldsymbol{I_3} \leftrightarrow$ Null

Finally, no more imitation is performed.

-----End of all the sessions-----

5 TOP-DOWN CONTROL BASED ON L_{md}

5.1 Attention mechanism

As mentioned above, the semantic understanding of human verbal suggestion makes a robot abstractly (i.e., conceptually) aware which matters and attributes involved in human demonstration should be attended to, and its pragmatic understanding provides the robot with concrete idea of real matters with real attribute values significant for imitation. More exactly, semantic understanding in $\boldsymbol{L_{md}}$ of human suggestion enables the robot to control its attention mechanism in such a top-down way that focuses the robot's attention on the significant attributes of the significant matters involved in human demonstration. Successively, in order for pragmatic understanding in $\boldsymbol{L_{md}}$ of human suggestion, the robot is to select the appropriate sensors corresponding with the suggested attributes and make them run on the suggested matters so as to pattern after the movements of human FAO implied by the locus formulas yielded in semantic understanding. ***That is to say in short,*** $\boldsymbol{L_{md}}$ ***expression suggests a robot what and how should be attended to in human demonstration and its environment.***

For example, consider such a suggestion as S12 presented to a robot by a human. In this case, unless the robot is aware of the existence of a certain box between the stool and the desk, such semantic understanding of the underlined part as (26) and such a semantic definition of the word 'box' as (27) are very helpful for it. The attributes A_{12} (Location), A_{13} (Direction), A_{32} (Color), A_{11} (Shape) and the spatial event on A_{12} in these $\boldsymbol{L_{md}}$ expressions indicate that the robot has only to activate its vision system in order to search for the box from the stool to the desk during the pragmatic understanding. That is, the robot can attempt to understand pragmatically the words of objects and events in an integrated top-down way.

(S12) Avoid <u>the green box between the stool and the desk</u>.

$(\exists x_1,x_2,x_3,x_4,p)(L(\_,x_4,x_1,x_2,A_{12},G_s,\_) \bullet L((\_,x_4,x_2,x_3,A_{12},G_s,\_))\Pi$

$L(\_,x_4,p,p,A_{13},G_s,\_)\Pi L(\_,x_2,Green,Green,A_{32},G_t,\_)$

$\wedge stool(x_1) \wedge box(x_2) \wedge desk(x_3) \wedge ISR(x_4)$ (26)

$(\lambda x)box(x) \leftrightarrow (\lambda x)L(\_,x,Hexahedron,Hexahedron,A_{11},G_t,\_)$

$\wedge container(x)$ (27)

(1) Data at t_1 **(2)** Data at t_2 **(3)** Data at t_3

Figure 13. Graphical interpretations of real motion data

Tom moved the right arm.
Tom raised the right arm.
Tom bent the right arm.
……

(a) Text for motion data from t_1 to t_2.

……
Tom lowered the right arm.
Tom stretched the right arm and simultaneously lowered the right arm.
……

(b) Text for motion data from t_2 to t_3.

Figure 14. Texts generated from real motion data

This top-down control of attention mechanism enables IMAGES-M can take in real human motion data through the motion capturing system in SDPU. For example, Figure 13 shows graphical interpretations of the real motion data taken in at the time point t_1, t_2 and t_3. These real data were translated via $\boldsymbol{L_{md}}$ into such texts as shown in Figure 14 by TPU. In this case, IMAGES-M's attention was guided by the suggestion S13 below.

(S13) Move your right arm like this.

5.2 Utilization of domain-specific knowledge

The linguistic knowledge $\mathbf{K_L}$ is employed exclusively for semantic understanding, consisting of syntactic and semantic rules and dictionaries. On the other hand, the domain-specific knowledge $\mathbf{K_D}$ is employed for pragmatic understanding and behaviouralization, containing all kinds of knowledge pieces acquired so far concerning the robot, the human and their environment. For example, the human body can be described in a computable form using locus formulas. That is, the structure of the human body is one kind of spatial event where the body parts such as head, trunk, and limbs extend spatially and connect with each other. The expressions (28) and (29) are examples of these descriptions in $\boldsymbol{L_{md}}$, reading that an arm extends from a hand to a shoulder and that a wrist connects a hand and a forearm, respectively.

$(\lambda x)\text{arm}(x)\leftrightarrow(\lambda x)(\exists y_1,y_2)\text{L}(\_,x,y_1,y_2,A_{12},G_s,\_)$
$\wedge\text{shoulder}(y_1)\wedge\text{hand}(y_2)$ (28)

$(\lambda x)\text{wrist}(x)\leftrightarrow(\lambda x)(\exists y_1,y_2,y_3,y_4)(\text{L}(\_,y_1,y_2,x,A_{12},G_s,\_)\bullet$
$\text{L}(\_,y_1,x,y_3,A_{12},G_s,\_))\wedge\text{body-part}(y_1)\wedge\text{forearm}(y_2)$
$\wedge\text{hand}(y_3)$ (29)

These descriptions are necessary for the robot to understand human action and text well enough to obtain an appropriate conception, eliminating such an anomalous one as is represented by S14 in a top-down way.

(S14) The left arm moved away from the left shoulder and the left hand.

Each of such human's/robot's motions (M_k) as 'walk' and 'bow' is given as an ordered set of its standardized characteristic snapshots (S_k) called 'Standard Motion' and defined by (30). In turn, a family (F_X) of S_ks is called 'Family of Standard Motions' and defined by (31), where the suffix 'X' refers to 'human (X=H)' or 'robot (X=R)'. The families F_H and F_R are contained in $\mathbf{K_D}$ and their members are employed for the default motions, namely, motions not specified in human suggestion or demonstration, during pragmatic understanding.

$S_k=\{M_{kS}, \ldots, M_{kE}\}$ (30)

$F_X=\{S_1, S_2, \ldots, M_N\}$ (31)

For example, the $\boldsymbol{L_{md}}$ expression of human walking in default is given by (32), reading that a human moves by his/her legs making his/her shape change monotonically from Walk_S to Walk_E.

$(\exists x,y,p_1,p_2,q_1,q_2)\ \text{L}(\_,y,x,x,A_{01},G_t,\_)\Pi$
$\text{L}(y,x,q_1,q_2,A_{12},G_t,\_)\Pi\ \text{L}(x,x,\text{Walk}_S,\text{Walk}_E,A_{11},G_t,F_H)$
$\wedge q_1\neq q_2\wedge\text{human}(x)\wedge\text{legs}(y)$ (32)

For another example, the $\boldsymbol{L_{md}}$ expression (33) is for the robotic motion of head shaking in default, reading that a robot affects its head in the Orientation (A_{14}), making its shape change monotonically from Shake_head_S to Shake_head_E. The shape values are given in a computable form general enough to reconstruct any human/robot motion in 3D graphics or so. Figure 15 shows an example of its interpretation in 3D graphics by PPU in IMAGES-M, which is also an example of cross-media translation from the text 'The robot shakes its head' into the animation.

$(\exists x,y,p_1,p_2)\text{L}(\_,y,x,x,A_{01},G_t,\_)\Pi\text{L}(x,y,p_1,p_2,A_{14},G_t,\_)\Pi$
$\text{L}(x,x,\text{Shake\_head}_S,\text{Shake\_head}_E,A_{11},G_t,F_R)$
$\wedge\text{robot}(x)\wedge\text{head}(y)$ (33)

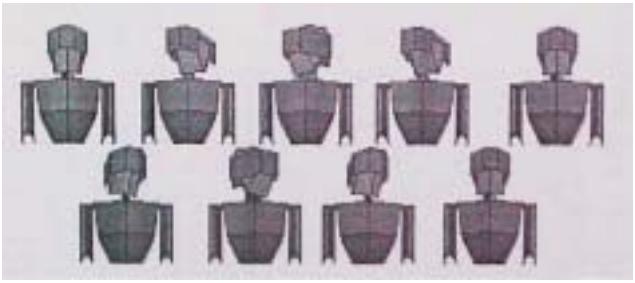


Figure 15. 3D animation of 'The robot shakes its head.'

5.3 Behaviouralization

The process for behaviouralization is to translate a conception (i.e., $\mathbf{C_i}$) into an imitation (i.e., $\mathbf{I_i}$) as a appropriate sequence of control codes for certain sensors or actuators in the robot to be decoded into a real behaviour by SDPU or ADPU in IMAGES-M. For this purpose, there are needed two kinds of core procedures so called 'Locus formula paraphrasing' and 'Behaviour chain alignment' as detailed below.

5.3.1 Locus formula paraphrasing

The attributes listed in Table 1 are essentially for human sensors or actuators and therefore the locus formula as $\mathbf{C_i}$ should be translated into its equivalent concerning the attributes specific to the robot's. For example, an atomic locus of the robot's 'Shape (A_{11})' specified by the human should be paraphrased into a set of atomic loci of the 'Angularity (A_{45})' of each joint in the robot. For another example, 'Velocity (A_{16})' for the human into a set of change rates in 'Angularity (A_{45})' over 'Duration (A_{35})' (i.e., A_{45}/A_{35}) of the robot's joints involved. These knowledge pieces are called 'Attribute Paraphrasing Rules (APRs)' [10] and contained in $\mathbf{K_D}$.

5.3.2 Behaviour chain alignment

Ideally, the atomic loci in the conception $\mathbf{C_i}$ (original or paraphrased) should be realized as the imitation $\mathbf{I_i}$ in a perfect correspondence with an appropriate chain of sensor or actuator deployments. Actually, however, such a chain as a direct translation of $\mathbf{C_i}$ must often be aligned to be feasible for the robot due to the situational, structural or functional differences between the human and the robot. For example of situational difference, in the simulation above, the robot must interpolate the travel from its initial location to the green box and the action to pick up the box. On the other hand, for example of structural or functional difference, consider the case of imitation by a non-humanoid robot. Figure 16 shows the action by a dog-shaped robot (SONY) to the suggestion 'Walk and wave your left hand.' The robot pragmatically understood the suggestion as '*I* walk and wave *my* left *foreleg*' based on the knowledge piece that only forelegs can be waved' and behaviouralized its conception as 'I walk *BEFORE* sitting down *BEFORE* waving my left foreleg' but not as 'I walk,

SIMULTANEOUSLY waving my left foreleg', in order not to fall down.

The procedure here [6, 12] is based on the conventional AI, where a problem is defined as the difference or gap between a 'Current State' and a 'Goal State' and a task as its cancellation. Here, the term 'Event' is preferred to the term 'State' and 'State' is defined as static 'Event' which corresponds to a level locus. On this line, the robot needs to interpolate some transit event X_T between the two events, 'Current Event (X_C)' and 'Goal Event (X_G)' as (34).

$$X_C \bullet X_T \bullet X_G \quad (34)$$

According to this formalization, a problem X_P can be defined as $X_T \bullet X_G$ and a task can be defined as its realization and any problem is to be detected by the unit of atomic locus. For example, employing such a postulate as (35) implying 'Continuity in attribute values', the event X in (36) is to be inferred as (37).

$$L(x,y,p_1,p_2,a,g,k) \bullet L(z,y,p_3,p_4,a,g,k) . \supset . p_3 = p_2 \quad (35)$$

$$L(x,y,q_1,q_2,a,g,k) \bullet X \bullet L(z,y,q_3,q_4,a,g,k) \quad (36)$$

$$L(z',y,q_2,q_3,a,g,k) \quad (37)$$

Figure 16. Robot's action to 'Walk and wave your left hand'

6 DISCUSSION AND CONCLUSION

The key contribution of this paper is the proposal of a novel idea of robotic imitation driven by semantic representation of human suggestion, where are hinted in the formal language $\boldsymbol{L_{md}}$ what and how should be attended to in human action as analogy of human FAO movement and thereby the robotic attention can be controlled in a top-down way. Without such a control, a robot is to simultaneously attend to tens of attributes of every matter involved in human action as shown in Table 1. This is not realistic, considering the difficulties in autonomous robotic vision understanding today. The author has a good perspective for the proposed theory of robotic imitation based on his previous work utilizing $\boldsymbol{L_{md}}$ for robot manipulation by text [6, 12]. This is one kind of cross-media operation via intermediate $\boldsymbol{L_{md}}$ representation [e.g., 6, 10, 12]. At my best knowledge, there is no other theory or system that can perform cross-media operations in such a seamless way as ours. This is due to the descriptive power of $\boldsymbol{L_{md}}$ enabling systematic organization and computation of spatiotemporal knowledge including sensation and action. Our future work will include establishment of learning facilities for automatic acquisition of word concepts from sensory data and multimodal interaction between humans and robots under real environments in order to realize the robotic imitation proposed here.

ACKNOWLEDGEMENTS

This work was partially funded by the Grants from Computer Science Laboratory, Fukuoka Institute of Technology and Ministry of Education, Culture, Sports, Science and Technology, Japanese Government, numbered 14580436 and 17500132.

REFERENCES

[1] A.Billard, 'Learning motor skills by imitation: biologically inspired robotic model', ***Cybernetics and Systems***, **32**, 155–193, (2000).

[2] A.Alissandrakis, C.L.Nehaniv, and K.Dautenhahn, 'Imitating with ALICE: Learning to imitate corresponding actions across dissimilar embodiments', ***IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans***, **32-4**, 482–496, (2003).

[3] J.Nakanishi, J.Morimoto, G.Endo, G.Cheng, S.Schaal, and M.Kawato, 'Learning from demonstration and adaptation of biped locomotion', ***Robotics and Autonomous Systems***, **47**(2-3), 79–81, (2004).

[4] J.M.Wolfe, 'Visual search in continuous, naturalistic stimuli', ***Vision Research***, **34**, 1187–1195, (1994).

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

[6] M.Yokota, 'Towards a universal language for distributed iIntelligent robot networking', ***Proc. of 2006 IEEE International Conference on Systems, Man and Cybernetics***, Taipei, Taiwan, (Oct., 2006).

[7] S.Coradeschi and A.Saffiotti, 'An introduction to the anchoring problem', ***Robotics and Autonomous Systems***, **43**, 85–96, (2003).

[8] E.Drumwright, V.Ng-Thow-Hing, and M.J.Mataric´, 'Toward a vocabulary of primitive task programs for humanoid robots', ***Proc. of International Conference on Development and Learning*** (*ICDL*), Bloomington,IN, (May, 2006).

[9] M.Yokota, 'An approach to integrated spatial language understanding based on Mental Image Directed Semantic Theory', ***Proc. of 5th Workshop on Language and Space***, Bremen, Germany, (Oct., 2005).

[10] M.Yokota and G.Capi, 'Cross-media operations between text and picture based on Mental Image Directed Semantic theory', ***WSEAS Trans. on INFORMATION SCIENCE and APPLICATIONS***, Issue 10, 2, 1541–1550, (2005).

[11] J.F. Sowa, ***Knowledge Representation: Logical, Philosophical, and Computational Foundations***, Brooks Cole Publishing Co., Pacific Grove, CA, (2000).

[12] M.Yokota, 'Integrated multimedia understanding for ubiquitous intelligence based on Mental Image Directed Semantic Theory', ***Handbook on Mobile and Ubiquitous Computing Innovations and Perspectives***, American Scientific Publishers, (in press).

[13] M.Egenhofer, 'Point-set topological spatial relations. Geographical Information Systems', 5,2 161-174 (1991).

[14] M.Nicolescu, M.J.Mataric´, 'Task Learning Through Imitation and Human-Robot Interaction, in Models and Mechanisms of Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions, Kerstin Dautenhahn and Chrystopher Nehaniv Eds., 407-424, (2006).

Short Presentations

Imitation in animals in lack of causal understanding?

Zsófia Virányi
Konrad Lorenz Institute for Evolution & Cognition Research, Altenberg, Austria

Various experimental results have shown that when causal information is available about a problem and a demonstrator's method to solve it chimpanzees prefer emulation and try to find their own (more efficient) method when presented with the same problem. At the same time it has also been suggested that they switch to imitation when causal structure of the problem and its demonstrated solution is unclear. It is questionable, however, whether more precise copying of the demonstrated actions in lack of knowledge about their relevance can be considered as imitation, or rather reflects emulation in animals which expect others' behaviour be efficient in lack of contradictory information.
In the present literature on human and non-human imitation two phenomena are described as unclear causal structure without clear differentiation between them: 1) in case of the above mentioned lack of full information of the observed action and its constraints efficiency of the action cannot be evaluated but can be assumed; 2) full information is available about the physical constraints and effects of the observed action, but they are in contradiction with the action itself (choice of the action cannot be explained by them).
Purpose of the poster is to draw attention to the need of differentiating between these two kinds of lack of causal understanding of social learning situations in order to avoid possible false comparisons between species and to make viable theoretical interpretations.

Selective imitation in dogs

F Range+, Zs Viranyi* §, L Huber+
+ Department for Behaviour, Neurobiology and Cognition, University of Vienna, Austria
§ Department of Ethology, Eötvös University, Budapest, Hungary
* Konrad Lorenz Institute for Evolution & Cognition Research, Altenberg, Austria

The transmission of cultural knowledge requires learners to identify what relevant information to retain and selectively imitate when observing other's skills. By one year of age human infants - without relying on language or theory of mind – already show evidence of this ability. They are able to interpret others' behavior as goal-directed, and as a result predict the most efficient action to achieve a goal within the constraints of a given situation. One situation in which human infants are thought to manifest this non-mentalistic inferential process is their selective imitation of goal-directed actions. For example, if a model demonstrates a head action when a hand action would be more efficient to turn on a light, infants imitate the head action only if its use during the demonstration cannot be explained by their hands being occupied, suggesting imitation by preverbal infants to be a selective, interpretative process (Gergely, Bekkering, Kiraly, 2002). However, the less effective action is only copied if the demonstration is accompanied by communicative cues targeted at the infants. Thus, early sensitivity to ostensive-communicative cues and the efficiency of goal-directed actions seem to be crucial prerequisites for such relevance-guided selective imitation (Csibra & Gergely, 2006). While this competence has been thought to be human-specific, here we show an analogue capacity in a non-human species, the domestic dog (Canis familiaris). In our experimental set-up, subjects watched a demonstrator dog pulling a wooden rod using an 'ineffective' paw action instead of using a mouth action usually preferred by dogs as was shown in a control group. In one group, using the 'ineffective' action was justified by the constraints of the situation e.g. the mouth of the model dog was occupied with a ball, whereas in the second group no constraints were present to explain the demonstrator's choice. In the first trial after observing the trained conspecific model, dogs imitated the non-preferred action only in the group where no constraints were present that could have explained the model's paw use. Consequently, dogs did not blindly copy the ineffective method, but demonstrated relevance-guided selective imitation like the infants in a comparable task.

Robotic Locust: who is my friend?

Shigang Yue
Brain Mapping Unit, Downing Site

To make a robot interact with human effectively, one important thing is to make sure it is able to recognise friendly and aggressive behaviours against it.

In this movie, we showed that it is possible for a robot to recognise these two different things around it. We equipted a khepera II robot with a pair of locust's inspired visual neural systems to see its surroundings, and a motor system to interpret the outputs of the visual system into behaviours.

The visual systems were based on lobula giant movement detector (LGMD) and decending contralateral movement detector (DCMD) in locusts. The visual-motor control was based on a motor system which may control locusts' directional jumping behaviours.

As shown in the movie, the robotic locust can recognise movements towards it by comparing the spikes from its two 'eyes'- escaping if it was an aggressive one, or just sitting there if it was a slow and gentle one, like a friend's movements.

The robotic locust always be able to run away from the fast approaching objects, which is often predators, regardless these objects' color, shape and materials.

We hope this move brings new inspiration ...

Object Affordances: Linking Sensory Motor Maps and Imitation

L. Montesano, M. Lopes, A. Bernardino, J. Santos-VIctor

The concept of affordance was introduced by Gibson as relation between an agent and the environment based on the agent's action capabilities. In this paper we argue that this concept (or knowledge representation) plays an important role as a bridge between sensory motor maps and higher cognitive capabilities such as imitation. Affordances encode relationships between actions, objects and effects and are at the core of basic cognitive capabilities such as prediction and planning. Within the framework of a general developmental architecture for social robots, we address the problem of learning affordances through the interaction of a robot with the environment as a key step to understand the world properties and interact socially. We present a general model for affordances using Bayesian networks. Actions, object features and effects form the nodes of the network and the affordances are implicitly encoded by the dependencies between these nodes. The amount of prior knowledge and the selected variables define different learning scenarios ranging from parameter tuning, which is the most common problem in the literature, to more general instances that also cope with feature selection and multiple actions. Since learning is based on a probabilistic model, it is able to deal with uncertainty, redundancy and irrelevant information present in real world. In addition to this, the model allows to directly use the acquired affordances to solve prediction, recognition and planning tasks. Using the affordances, the robot is able to imitate a human based on the perceived effects and its knowledge about its own action capabilities. We demonstrate successful affordance learning on a humanoid robot interacting with objects and apply the acquired knowledge in simple imitation games.