Hierarchies of Coupled Inverse and Forward Models for Abstraction in Robot Action Planning, Recognition and Imitation

Matthew Johnson & Yiannis Demiris
*Imperial College London
Exhibition Road, London, SW7 2BT
{matthew.johnson,y.demiris}@imperial.ac.uk

Abstract

Coupling internal inverse and forward models gives rise to on-line simulation processes that may be used as a common computational substrate for action execution, planning, recognition, imitation and learning. In this paper, multiple coupled internal inverse and forward models are arranged in a hierarchical fashion, with each level of the hierarchy interacting with other levels through top-down and bottom-up processes. Through experiments involving imitation of a human demonstrator performing object manipulation tasks, this architecture is shown to equip a robot with a multi-level motor abstraction capability. This is then used to solve the correspondence problem in action recognition. The architecture is inspired by biological evidence.

1 Introduction

Research has shown the direct involvement of the human motor system when observing, as well as imitating, actions performed by others (Meltzoff and Decety, 2003). This, along with the neuroscientific discovery of mirror neurons in area F5 of the macaque monkey premotor cortex, which respond when both performing and observing the same action (Rizzolatti et al., 2002), has led to the proposition of a mirror system underlying the recognition and understanding of behaviour (Fadiga and Craighero, 2003). This system is compatible with the simulation theory of mind-reading (Gordon, 1999), and connections have been made between the two (Gallese and Goldman, 1998). Much progress has been made in building artificial models of the mirror system, particularly using internal models (Demiris, 1999). Such models have been deployed onto robots, so as to investigate the practical aspects of using the simulation theory to understand and imitate the behaviour of other robots and of humans (Demiris and Johnson, 2003, 2004). Experiments with this approach have demonstrated that recognising the actions of a human requires a robot to apply a motor abstraction capability to observed actions, otherwise the recognition is impossible due to differences in human–robot morphology, and the much greater size of the human action space compared to that of the robot (Johnson and Demiris, 2004). In this paper that abstraction is achieved through modeling the motor system as a hierarchy of multiple coupled internal inverse and forward models.

2 Background

2.1 Inverse Models

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 (Narendra and Balakrishnan, 1997; Wada and Kawato, 1993). In the control 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 the architecture described in this paper there are multiple inverse models, used at different levels of a hierarchical action execution and recognition system. When using multiple inverse models, each inverse model is considered valid for a specific goal or set of goals; that is, it can be used to achieve those goals. Thus, the purpose of an inverse model can be defined in general terms by the region of the goal space for which it is valid, and in specific terms at a single point in time by a particular goal taken from
within that region. For example there may be an inverse model “grasp object”, whose purpose is to be able to grasp a variety of possible objects. The further specification of the goal, such as specifying which object is to be grasped, may be supplied to the inverse model as a goal parameter.

There are situations in which an inverse model may or may not generate output. These situations are represented in the inverse model by the following states:

- If an inverse model is producing output from a current state and set of goal parameters, then it is in the state of executing.
- If, through comparison, the inverse model calculates that the current state is sufficiently close to the specified goal state, then no action is required. In this situation, the inverse model is complete.
- The inverse model may be presented with a current state that renders it unusable, as regards its purpose. The inverse model is then ineligible. An example would be a “Place object on table” inverse model, when there is no object.
- Although the current state may make the inverse model eligible for use, there may be a specified goal parameter for which the inverse model cannot produce any action that will result in it becoming complete. In this case, the inverse model is not applicable. An example would be the “Place object on table” inverse model when the object placement location has been obstructed.

The inverse model states defined above are considered binary states.

The inverse models described in this paper are not equipped with explicit initial knowledge as to the region of the goal space for which they are applicable. Instead, the inverse models determine whether or not they are capable of achieving a specific goal through an ongoing, active, simulation process, which performs action planning and results in action generation. This simulation planning requires the use of a forward model.

2.2 Forward Models

The generic forward model takes as input the current state of the system and the control signals acting on the system, and offers as output a prediction as to the next state of the system (Jordan and Rumelhart, 1992). In this 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 (Wolpert and Kawato, 1998; Wolpert et al., 2003). When coupled to an inverse model, a forward model receives the action output from the inverse model through an efference copy. The forward model then generates a prediction of the state that would result, if the action was to be performed. This prediction can then be used for action planning and action recognition, as described in section 3 below.

2.3 Abstraction in Recognition

The architecture described here achieves action recognition by matching internally generated actions to observed external actions. In doing so, it is solving the correspondence problem (Nehaniv and Dautenhahn, 2002; Alissandrakis et al., 2002). When using robots to recognise and imitate actions performed by a human, solving the correspondence problem is made more complicated by the difference in morphology. This difference can lead to considerable disparities between the actions the robot would use to accomplish a task, and the actions the human uses to accomplish the same task in a demonstration. If the difference in morphology is small, i.e. if the robot is humanoid but with fewer degrees of freedom, then the robot can be equipped with a human motion model for action generation, which will bring the robot’s actions closer in nature to that of the human demonstrator (Simmons and Demiris, 2004).

However, if the robot’s morphology is so dissimilar to that of a human that it cannot produce human-like actions, then this is a direct problem for using simulation theory for action recognition in robots. To address this issue, the motor system is developed as a hierarchical architecture, in which actions are prepared before execution using inherently more abstract simulation processes at higher levels of the hierarchy, a strategy similar to that used in (Haruno et al., 2003). Motor abstraction for successful recognition of observed human actions is then accomplished by using the higher levels of the hierarchy in a simulation theory approach.

3 The Hierarchical Architecture

3.1 Overview

The hierarchy is constructed using multiple coupled inverse and forward models. Figure 1 gives an overview of a hierarchy of $K$ levels. The lowest level of the hierarchy contains a set of primitive inverse
models $I_p$, which generate motor commands $M_t$ at each timestep to directly activate motor units (Benviega and Atkeson, 2002). The forward model in this level is a forward kinematics model of the robot, and thus offers predictions as to the trajectory that results from executing a motor command.

Higher-level inverse models generate actions that are sent down to the lower levels of the hierarchy for further interpretation and elaboration. Actions at higher levels are thus a more abstract representation of the eventual motor behaviour of the robot. The higher-level forward models offer predictions as to the outcomes and internal states of the inverse models in the lower levels that would result from the action, when it is interpreted in the level below. For example, an inverse model “grasp object” will have an outcome state “holding object = true”, and the “gripper close” inverse model will then become ineligible for use. Thus the coupling of the high level forward and inverse models provides a simulation capability that is abstract over spatial and temporal trajectory, and which can be used for abstraction in action planning and recognition.

### 3.2 Action Representation

At the lowest level of the hierarchy, the primitive inverse models generate actions that are motor commands, meaning that they directly stimulate their intended motor units in order to realise the given action. At higher levels, inverse models generate actions that are represented by action graphs and goal parameter vectors. These actions require further elaboration at lower levels to enable final execution.

Action graphs are constructed as directed acyclic graphs, in which the nodes are inverse models and the edges specify the sequence of inverse model execution. These inverse models may produce actions that are themselves constructed as action graphs and goal parameter vectors, which are then passed on to the lower level of the hierarchy.

The recursive formulation of the action graph for action representation allows for a multi-level hierarchy of inverse models in action generation. An action is performed by traversing the action graph. The inverse models encountered are executed in the lower levels with the goal parameters supplied by the goal parameter vector until they are complete, and then the traversal continues.

An action graph is represented throughout the architecture by its adjacency matrix, denoted $\psi$ (Jain and Krishna, 2003). To construct $\psi$, the $N$ inverse models in the lower level of the hierarchy are enumerated $1, \ldots, n$, so as to index the rows and columns of $\psi$ during its construction. To demark the beginning and end of an action, and to facilitate computation and processing, the marker nodes start and end are introduced. $\psi$ then becomes an $(N + 2) \times (N + 2)$ matrix. The adjacency matrix is constructed such that if there is a directed edge from node $i$ to node $j$ ($i \rightarrow j$) then the matrix element in the $i^{th}$ column and $j^{th}$ row of $\psi$ ($\psi_{ij}$) equals one ($\psi_{ij} = 1$), otherwise it is set to zero ($\psi_{ij} = 0$). Thus, when parsing the matrix, an entry of “1” indicates that there is an edge from the node specifying the column to the node specifying the row, and an entry of “0” indicates no connection. When executing an action using $\psi$, the matrix is interpreted in a breadth-first manner, so that all the inverse model nodes leading to a single node must be completed before moving on to executing that subsequent node. This allows an action to be comprised of many parallel-executing components. An example of an action graph is given in Figure 2(A), and an example of $\psi$ is given in Figure 2(C).

The goal parameter vector, denoted $\lambda$, has an entry for each of the $N$ inverse models enumerated as for the action graph. If a particular inverse model requires no goal parameters, then its respective entry in $\lambda$ remains zero.

### 3.3 Efferent Signals

When a higher-level inverse model generates an action, that action is sent in the form of an adjacency matrix and goal parameter vector as an efferent signal, to the level beneath in the hierarchy. The subsequent evaluation of the ensemble $\{\psi, \lambda, I\}$ of the adjacency matrix $\psi$, the goal parameter vector $\lambda$, and the set $I$ of inverse models, results in the generation of more specific actions, and those actions are propagated all the way down the hierarchy, until the action becomes a motor command $M_t$ and is eventually realised in the motor units.

### 3.4 Afferent Signals

Proprioceptive information for joint configurations, and exteroceptive information regarding objects in the environment, are continually provided by sensor units. This information is arranged into the current state vector $S_t$ and is sent up through the hierarchy as an afferent signal. Every level of the hierarchy receives this signal. For higher levels, the state information is supplemented by the status of the inverse models in the previous level, i.e. whether those inverse models are complete, eligible, applicable, or executing. Along with the efferent signals from the
level above, this afferent flow of status information provides for reciprocal connections between the levels of the motor system.

### 3.5 Simulation Processes

The dashed lines in Figure 1 mark the feedback generated from the closing of two simulation loops. These simulation loops may be used for action execution, planning, recognition, and learning, depending on the requirements of the robot.

#### 3.5.1 Inner Loop

The inner simulation loop is used for planning and modulating an on-going action during action generation. The inverse model generates multiple action hypotheses that it postulates will achieve the specified goal parameters. The action hypotheses are tested on the forward model, resulting in predicted states that are sent back to the inverse model. The inverse model can then use these predicted states in substitution for the current state, creating a simulation process that allows it to plan actions into the future, by searching the possible action space. Through comparison with the goal parameters, the inverse model converges to an action solution. There may be many potential action solutions that accomplish a given goal. The most appropriate solution at any given time is selected by a winner-takes-all mechanism, on the basis of the smallest action-graph depth, and sent to the level below. All the levels perform the same simulation process continually, and in parallel. The result is a distributed on-line hierarchical control model that directly and indirectly modulates an action as it unfolds.

If, through the inner-loop simulation process described above, an inverse model determines that it is unable to achieve its goal, then this “not applicable” state is signalled as part of the overall state of the inverse models in that level (other states are complete, eligible, and executing). The afferented robot state information is supplemented by this inverse model state information as it reaches each higher level. The combined state information is then used in the outer simulation loop.
3.5.2 Outer Loop

The outer loop is a prediction-comparison process. The forward model produces a prediction $\hat{S}_t$ as to the result of the supplied action solution, and this is buffered by the delay component $D$, before comparison with the actual resulting state, $S_t$. The resulting prediction error $P_e$ may be used both for action generation and learning of forward and inverse models when the supplied current state is the agent’s own (Haruno et al., 2003), or action recognition and imitation learning, when the current state is that of an observed actor (Demiris and Hayes, 2002; Demiris and Johnson, 2003). In this architecture, the prediction error is calculated as being the sum over the $n$ state elements, of the absolute difference between the predicted state and the actual state:

$$P_e = \sum_{i=1}^{n} |S_{t,i} - \hat{S}_{t,i}|$$  \hspace{1cm} (1)

3.6 Recognition and Imitation

The same arrangement of structures, as shown in Figure 1, is used for action recognition as well as execution. In recognition, the state input to the architecture is not taken from the robot, but is derived from visual observation of the demonstrator. All the inverse models in every level of the hierarchy that are “eligible” for execution, and not “complete”, are then executed in parallel. The inverse models in a particular level compete with the other inverse models in that level for confidence, which is awarded at each time step to inverse models that match well with the perceived action. A winner-takes-all selects the inverse model with the highest confidence at any point in time as being the recognised action. The robot’s motor hardware is taken off-line to prevent physical “mirroring” of the perceived action, by inhibiting the motor commands generated by the primitive inverse models in the lowest level of the hierarchy. When recognition is complete, imitation may proceed by executing the observed action.

3.7 Confidence Calculation

3.7.1 Lowest Level

The inverse models compete for confidence. At each timestep, the inverse model with the lowest prediction error $P_e$ is rewarded, and the rest of the inverse models are punished. The inverse model with the lowest prediction error has its confidence $C_t$ rewarded as follows:

$$C_t = \begin{cases} C_{t-1} + \frac{1}{\epsilon} & \text{if } P_e < \epsilon \\ C_{t-1} + \frac{1}{P_e} & \text{otherwise} \end{cases}$$  \hspace{1cm} (2)

The other inverse models have their confidences punished, according to:

$$C_t = \frac{C_{t-1}}{2}$$  \hspace{1cm} (3)

Initial confidences are zero for all inverse models. In the following experiments, $\epsilon$ was chosen to be 0.04.

3.7.2 Higher Levels

The forward models predict the outcomes and internal states of the lower-level inverse models that are the components of the action input. Thus, the higher-level inverse models are rewarded when the prediction error $P_e$ is less than $\epsilon$, and their confidences are reset when they become complete:

$$C_t = \begin{cases} C_{t-1} + \alpha & \text{if } P_e < \epsilon \\ 0 & \text{if inverse model is complete} \end{cases}$$  \hspace{1cm} (4)
Figure 3: Confidence levels of primitive inverse models in the lowest level of the hierarchy during a demonstration of picking up an object and placing it back on the table. The sequence of movements is: move to object → move object away from table → move object to table → move away from object. The confidence values have been normalised at each time step.

As the prediction error is less than $\epsilon$ only at specific times, the confidence is never punished and the inverse models do not compete for confidence. Initial confidences are zero for all inverse models. In the experiments that follow, $\alpha$ was chosen to be 10.

4 Implementation

To demonstrate the architecture, it was implemented in a two-level hierarchy on a robot in an experimental scenario involving the recognition and imitation of object manipulation actions performed by a human demonstrator.

The lower level of the hierarchy was populated with six primitive inverse models, “gripper open”, “gripper close”, “move to object”, “move away from object”, “move object to table”, and “move object away from table”. The higher level was equipped with the inverse models “grasp object” and “place object”, both of which accomplished their goals by combining the low-level primitives into action graphs. To simplify the implementation, only one object and one table were used, restricting the goal parameter space.

4.1 Robot Platform

The Peoplebot is equipped with a Canon VCC4 pan-tilt-zoom (PTZ) camera, two degrees of freedom gripper, and sonar and infra-red sensors. In these experiments, the camera was used as the main tracking and range-finding sensor. The sonar and the infra-red sensors were not used. All processing was done in real-time, with one full iteration of the architecture’s mainloop executing in 0.5 seconds. The software was written in C++ for an AMD Athlon 64, which controlled the robot remotely over a wireless ethernet link.

4.2 Visual Systems

The visual tracking of the object and the hand was accomplished using the CAMShift algorithm (Bradski, 1998), working on a hue and saturation histogram back-projection of camera images taken at a pixel resolution of 640 × 480 and at 2 frames per second. The low frame rate was deliberately chosen to reduce noise in the visual signal. The ARToolkit (Billinghurst et al., 2001) was used to determine the robot’s position relative to the table, as stereo vision was not available on the robot. Depth information was thus obtained by affixing an 8 cm × 8 cm marker to the table’s midpoint.

5 Experiments

The object manipulation actions chosen for the experiments were the common tasks of picking an object up from a table, and placing an object onto a table. These behaviours are well suited to the robot used, an ActivMedia Peoplebot, with its mobile platform and gripper assembly.

The robot was positioned facing a table, upon which was placed an object that was readily manipulable by both the robot and the human demonstrator. In these experiments, the object used was a tub. The initial robot-table distance was 1 m, sufficient for the robot’s camera to view the entire scene, including the table, object, and the hand of the demonstrator as she moved to place or pick up the object. The demonstrator was unfamiliar with the operational details of
the architecture, and was instructed when to start the demonstration. If the robot recognised the demonstrated action then it performed the action for itself, completing the cycle of imitation.

6 Results

Figures 3 and 4 show typical results from the experimental trials. Figure 3 shows the confidence levels of four primitive inverse models in the lowest level of the hierarchy during a demonstration of picking up and placing the object (the primitive inverse models shown in this graph are “Move to object”, “Move object away from table”, “Move object to table”, and “Move away from object”). The architecture achieves successful recognition, ascribing high confidence levels to the primitive inverse models that generate trajectories that match with the observed actions. The progression of the confidence values shows the competition between inverse models during transitional stages of the action, where one inverse model builds up confidence at the expense of the others (iterations 12-14, 24-26, and 30-31). The duration of a recognised action can be seen as the length of time that the confidence level for a particular primitive remains at 1.

Figure 4 shows the confidence levels of the two inverse models in the higher level of the hierarchy. The peaks in the confidence clearly demark the “grasp object” action and the subsequent “place object” action. The higher-level inverse models do not match on action trajectory, but on subgoals during an action, resulting in a more abstract recognition that clearly distinguishes different observed behaviours.

7 Discussion

For large numbers of inverse models in any given level of the hierarchy, an adjacency matrix becomes a memory-inefficient means of action representation. The computational cost of adding inverse and forward models is therefore less than the overall memory cost. However, due to the directed nature of the action graphs, the matrix $\psi$ is sparse, and can be efficiently managed through the use of look-up tables. It is expected that on modern computers the system could handle up to and beyond a hundred inverse and forward models.

Although the abstraction architecture is capable of recognising actions performed in different ways, the visual system is sensitive to the speed at which the actions are performed. This results in situations where recognition may not be successful. If the demonstrator moves too slowly, then noise in the visual system overcomes the movement signal and lower-level recognition fails, although higher-level recognition may succeed. Recognition at all levels fail if the human performs the movement too fast for the architecture to extract a reasonable signal.

Acknowledgments

This research was funded by the UK’s Engineering and Physical Sciences Research Council (EPSRC) project GR/S11305/01, and the Royal Society. The authors would like to thank the BioART team, and also Mercedes Lahnstein and Tania Katzouraki for their invaluable support and assistance.
References


