Perspective Taking Through Simulation

Matthew Johnson **Yiannis Demiris**
Department of Electrical and Electronic Engineering
Imperial College London
Exhibition Road, London, SW7 2BT
{matthew.johnson, y.demiris}@imperial.ac.uk

Abstract

Robots that operate among humans need to be able to attribute *mental states* in order to facilitate learning through imitation and collaboration. The success of the *simulation theory* approach for attributing mental states to another person relies on the ability to *take the perspective* of that person, typically by generating *pretend states* from that person's point of view. In this paper, internal inverse and forward models are coupled to create simulation processes that may be used for mental state attribution: simulation of the visual process is used to attribute perceptions, and simulation of the motor control process is used to attribute potential actions. To demonstrate the approach, experiments are performed with a robot attributing perceptions and potential actions to a second robot.

1. Introduction

One of the most widely researched theories of how we attribute mental states to others is the *simulation theory* (Nichols and Stich, 2003, Gordon, 1999, Gallese and Goldman, 1998). By this theory, people attribute mental states using their own mental processes and resources as manipulable models of other people's minds, taken *off-line* and used in simulation with states derived from *taking the perspective* of another person.

Much recent research has been done in applying simulation theory techniques to robotics, most notably for *action recognition* and *imitation learning*, using internal *inverse and forward models* (Demiris and Johnson, 2005, Demiris and Johnson, 2003). In this paper, simulation of perception is used in conjunction with simulation of actions for the purposes of perspective taking. Simulation of perception is achieved by introducing *internal inverse and forward visual models*, which are models of the visual perception process. Simulation of actions is achieved by coupling *inverse control models* to *forward control models*, which are models of robot control dynamics. To integrate perception simulation with action simulation, *internal spatial models* are introduced, which model spatial relationships between observed objects within an environment.

The combination of these internal models results in an architecture that uses simulation in order to attribute perceptions and potential actions to observed subjects through perspective taking. This architecture is demonstrated in robotic experiments, with a robot attributing perceptions and potential actions to a second robot.

2. Background

2.1 Internal Forward Models

In the architectures developed in this paper, we use two kinds of internal forward model: *forward control models*, which are used in action simulation; and *forward visual models*, which are used in perception simulation. Figure 1 and Table 1 present a comparison of internal visual and control models.

2.1.1 Forward Control Models

Forward control models (often referred to as just 'forward models' (Demiris, 1999)) 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*. When used in control theory, the forward model is used to model the causal dynamics of a *plant*, and the inputs are the plant's state and the *control signals* acting on the plant. The concept is readily extended to robotics; in this case, the states are the robot's joint kinematics and dynamics, and the control signals are *motor commands*. Predictions as to the next state of the robot may be used for predictive control, and learning (Miall and Wolpert, 1996).

Forward models may be extended to predict the *outcomes* of actions, in terms of the state of the environment and the objects within it that are affected by the action. The outcome predictions produced by forward models can be used in action planning and recognition (Johnson and Demiris, 2005).

In many systems the state space and inherent dynamics are so extensive that it is impossible to di-

Figure 1: The generic inverse (I) and forward (F) model; the same input-output relation is used for both internal *control* and internal *visual* models. An explanation of the inputs and outputs is given in Table 1.

Label	Control Models	Visual Models
S	Current state	Visual object description
u	Control signal	Visual scene
G	Desired state	Desired object state
$\hat{S}$	Predicted state	Object state

Table 1: Comparison of internal control and visual models. Internal control models function on a temporal dimension to generate control signals and state predictions; internal visual models function on a spatial dimension to process and construct visual scenes.

rectly construct a generic forward model. Instead, learning techniques are used (Dearden and Demiris, 2005, Jordan and Rumelhart, 1992) or multiple forward models are produced, each one specific to a certain dynamic or sector of the state space.

2.1.2 Forward Visual Models

A forward *visual* model is defined as being a model of the visual process. The generic forward visual model takes as input a visual scene, along with visual object or feature descriptions, and outputs state information regarding those objects and features, such as object positions and orientations.

There may be *multiple* forward visual models in a system. In this situation, each forward visual model is used for the visual processing of specific features in the supplied visual scene.

2.2 Internal Inverse Models

Internal *inverse* models are the complement of internal forward models. They attempt to invert the process modeled by the respective forward model.

2.2.1 Inverse Control Models

An inverse control model, referred to in the control literature as an *inverse model* (Narendra and Balakrishnan, 1997), takes as input a current system state and a desired future system state, and produces as output the system dynamics that will cause the system to move from its current state to the desired future state. In the control literature,

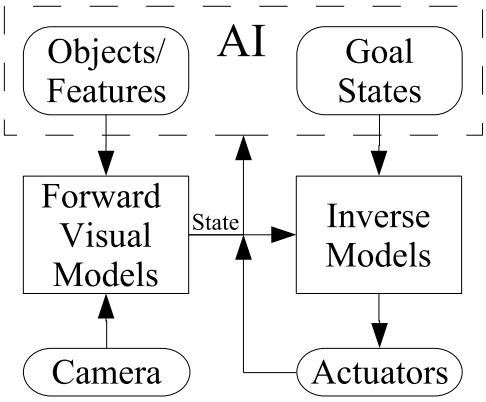

Figure 2: Forward visual models coupled to inverse control models for visuomotor transform and visual servoing.

inverse models are referred to as *controllers* and their outputs are control signals; when applied to robotics, the current state is the state of the robot and its environment, the desired future system states are *goal states*, and the outputs are *motor commands*.

Inverse models have *internal states* (Johnson and Demiris, 2005), that are used in action generation and action simulation:

- If an inverse model is producing control 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 shelf" inverse model, when there is no object.
- 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. An applicability level of zero 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 *confidence* level of an inverse model is used for *action recognition*, and is a measure of how well the actions generated by that inverse model match with an action under observation (Demiris, 1999).

Confidence and *applicability* are scalar metrics, generated through *simulation–comparison processes*, that may

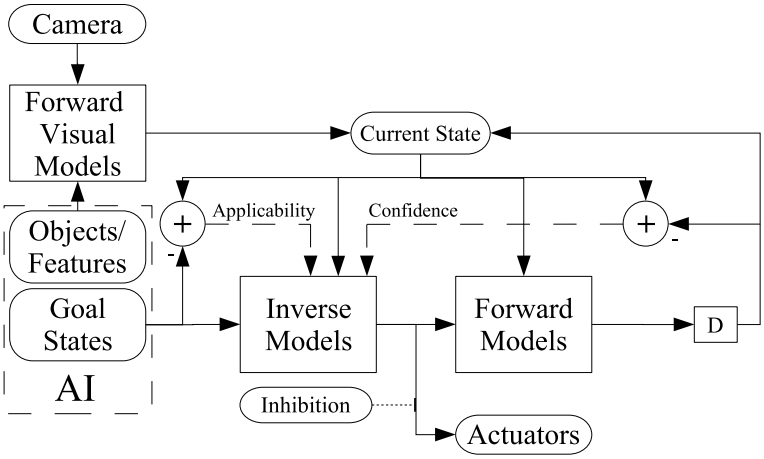

Figure 3: A forward model coupled to an inverse model to produce a simulation loop. This loop is used in *simulation–comparison* processes to generate the inverse causal model states *applicablity* and *confidence*.

take any value. The other states are considered binary states, and are updated during both action generation and simulation.

2.2.2 Inverse Visual Models

The *inverse visual model* inverts the visual process by taking as input visual descriptions of objects and features, their desired states i.e. positions and orientations, and constructing a visual scene as output. Because the inverse visual model is the inverse of the visual process, the constructed visual scene includes spatial perspective effects.

Because of resource constraints for construction of the visual image, there is often only one inverse visual model in a system.

2.3 Internal Spatial Models

An environment, at any given moment of time, will contain objects and other entities that possess distinct spatial co–ordinates and spatial relationships with each other. An *internal spatial model* is a manipulable model that maintains knowledge of those relationships.

Rather than being a model of a *process*, an internal spatial model is a model of spatial *associations*. The *forward* spatial association takes as input an environment description and an object description, and determines the *location* and *orientation* of that object within the environment. The *inverse* spatial association, which is not used in this paper, takes object descriptions and object locations and determines the *environment* in which they belong. Several spatial models may exist, each defining spatial relationships for specific environments or objects.

Internal spatial models may be recruited by simulation of perception, or simulation of action, or both.

3. Perspective Taking Using Internal Models

Forward visual models can be coupled to inverse control models for *visuomotor transform* and *visual servoing* (Figure 2). In this arrangement, the forward visual models extract state information about objects or features in the visual scene, and supply this to the inverse models, which control the actuators. The actuators themselves provide proprioceptive state information which is added to the overall state information. The state information is used to determine the goal states for the inverse models and the desired objects or features for the forward visual models.

Action simulation can be achieved by adding *forward models* to this architecture. The forward model is coupled to the inverse model through *efference copy*. This is shown in Figure 3. Action simulation may be used for determining which inverse models can be used for interacting with observed objects, and for planning actions into the future.

By *adopting the perspective* of an observed subject, it is possible to use this action simulation loop in order to determine what actions are available to that subject given its environment, and also to recognise the actions that the subject is performing. This is done by having the inverse models accumulate *applicability* and *confidence* through *simulation–comparison loops*. To achieve the perspective taking, internal visual, control, and spatial models are arranged in a perception-action simulation architecture. Figure 4 shows the arrangement. A forward visual model, dedicated to extracting the eye gaze of the observed subject, works on the input visual image in order to determine a *viewpoint transformation*. The internal spatial model, updated with extracted visual information, is manipulated according to the view-

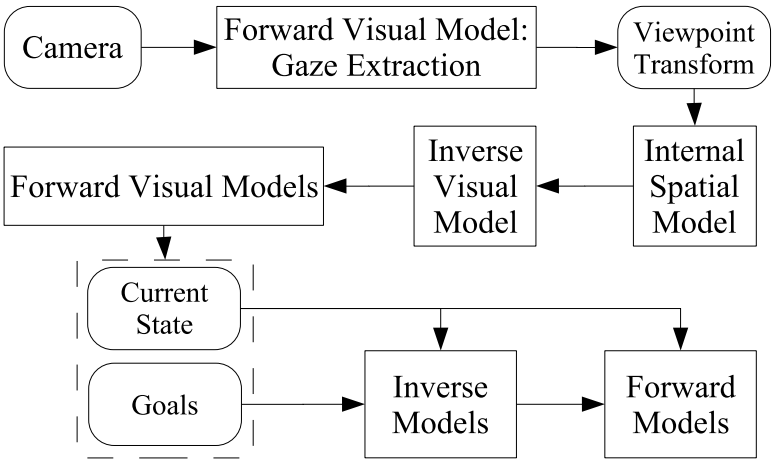

Figure 4: The internal models involved in attribution of perception and action through simulation.

point transformation and used to provide state input to the inverse visual model. The inverse visual model constructs an image that describes the visual scene *as the observed subject would see it*. The constructed image is then processed by forward visual models in order to extract visual state information and thus attribute perceptual states to the observed subject.

The attributed perceptual states are then fed into the inverse models. The inverse models are all executed in parallel. The control signals generated by the inverse models are inhibited from reaching the actuators and instead are sent to the forward models, in order to produce an *action simulation loop*. Through action simulation, the level of applicability of each inverse model is determined. Inverse models with high levels of applicability are attributed to the observed subject as being *potential actions*.

4. Experiments

The architecture shown in Figure 4 was implemented on an ActivMedia Peoplebot for perspective-taking experiments. The experiments involved the Peoplebot observing a second, "target", Peoplebot facing a table upon which were placed two graspable objects. The purpose of the experiment was for the Peoplebot to attribute perceptual states and potential actions to the target, by taking its perspective, and to compare this with results taken from the robot's first-person perspective.

4.1 Experimental Setup

Figure 5 shows a plan view of the experiment. The target robot was placed facing a table at 1m distance. The observing robot was placed at right-angles and 1.5m distance to this setup, in order for it to see both the target robot and the table and objects. Two objects were placed on the table, a cylindrical tub and a cuboid block. The objects were placed side-by-side such that the observing robot could see both, but the target robot could see only the cuboid block, with the cylinder being obscured.

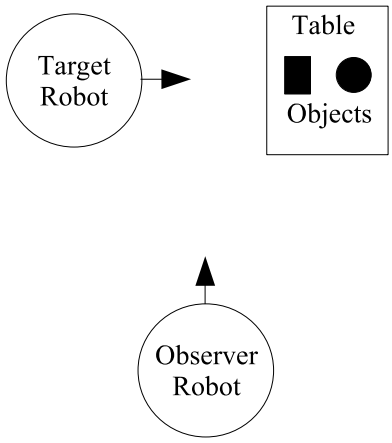

Figure 5: Plan view of the experimental setup. The arrows indicate robot camera direction. The cuboid and cylindrical objects are placed on the table such that the observer robot can see both but the target robot can only see the cuboid object.

5. Implementation

5.1 Robot Platform

The observing Peoplebot was 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, with the sonar and the infra-red sensors not used. All processing was done in real-time, with one full iteration of the architecture's main-

loop executing in 0.1 seconds. The software was written in C++ for a 2 GHz AMD Athlon 64, which controlled the robot remotely over a wireless ethernet link.

The "target" Peoplebot did not have a PTZ camera, and was instead equipped with a Philips USB webcam connected to a mini-ITX computer onboard the robot. This camera was used to determine the scene from the robot's point of view, for comparison with the image constructed by the observing robot's inverse visual model when taking the target robot's perspective.

5.2 Forward Visual Models

Forward visual models were implemented using the CAMShift algorithm (Bradski, 1998) and the ARToolkit (Billinghurst et al., 2001). The CAMShift algorithm worked on a hue and saturation histogram back-projection and was used for tracking of the manipulable objects. The ARToolkit was used to determine the robot's position relative to the table, objects, and target robot, as stereo vision was not available. To aid in the extraction of 3D location information, symbols from the ARToolKit were thus attached to the table, objects, and target robot.

5.3 Inverse Visual Models

To construct visual scenes from visual object descriptions and locations, the inverse visual models used the OpenGL graphics library[1]. OpenGL uses geometric descriptions and visual feature descriptions of colour and texture to construct a visual image using specified camera parameters such as field-of-view. In keeping with the simulation theory approach, the camera parameters chosen were kept as close to the observing robot's Canon VCC4 as possible.

5.4 Internal Spatial Models

The spatial models used in the experiment were comprised of the geometrical shapes of the cuboid and cylindrical objects, the table, and the target robot. The spatial relationships between these shapes was not coded into the spatial model, but were instead extracted from the visual scene using a forward visual model implemented using the ARToolkit.

5.5 Forward Control Models

Two types of forward model were implemented:

1. Forward models of the robot - these were implemented using numerical integrators, working on a kinematics model of the robot. This formulation was made possible by linking the motor commands produced by the inverse models to derivatives of the state space, e.g. a motor command represented a joint velocity.
2. Forward outcome models - these forward models generated predictions as to the expected outcome of the action, should the respective inverse model be executed to completion.

[1] http://www.opengl.org

5.6 Inverse Control Models

To demonstrate the attribution of actions through simulation, the observing robot was equipped with two inverse models, "move to object 1" (the cuboid) and "move to object 2" (the cylinder). At any point in time, each inverse model was *executing*, *complete*, or *ineligible* for the current state, and had a level of *applicability* as regards its goal.

The inverse models determined whether or not they were *complete* at each timestep by calculating the sum over the M state elements, of the absolute distance between the current state S_t and the goal parameters λ (the goal parameter being the location of either object 1 or object 2):

$$S_d = \sum_{i=1}^{M} |\lambda_i - S_{t,i}| \tag{1}$$

When S_d was less than a completion threshold ϵ, the inverse model became *complete* and did not generate motor commands even when instructed to execute.

To determine *eligibility*, each inverse model was provided with a set of state vectors Υ for which it was *ineligible* for execution. At each timestep, an inverse model calculated its eligibility through comparison of the current state with each element of this set. If the current state was within this set ($S_t \in \Upsilon$), then the inverse model was ineligible and did execute.

The inverse models constantly simulated action generation using the supplied goal parameters. During this simulation process, the distance between the current state and the goal was calculated through comparison using equation 1, and the *applicability* A_t was then accumulated for the n^{th} simulation iteration according to:

$$A_{t,n} = \begin{cases} 0 & \text{at } n = 0 \\ A_{t,n-1} + \dfrac{1}{S_d} \times \dfrac{1}{n} & \text{otherwise} \end{cases} \tag{2}$$

The applicability was re-calculated every time step, using the current state S_t as the initial starting state for the simulation. The simulation process continued until either the inverse model became *complete* (in simulation) or until the number of simulation iterations exceeded the iteration limit $N = 1000$. The resulting applicability level determined how useful each inverse model was for achieving its goal. In the experiment carried out in this

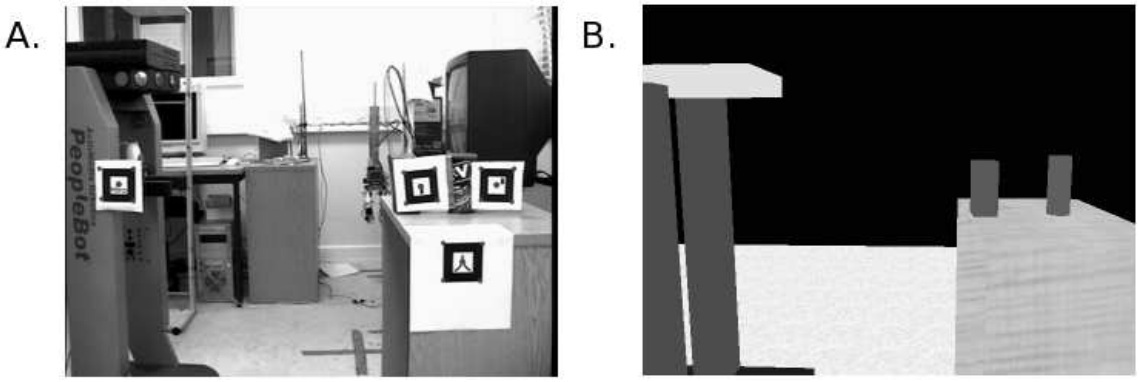

Figure 6: (A) The visual scene from the observing robot's point of view. (B) The internally generated image of the observing robot's point of view.

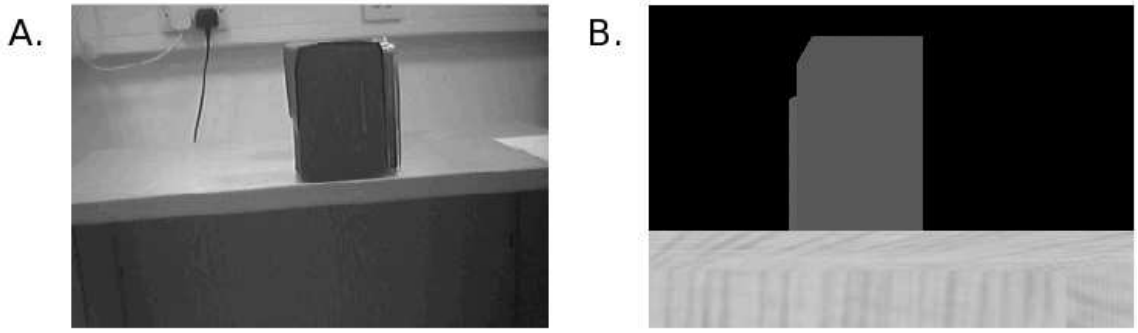

Figure 7: (A) The visual scene from the target robot's point of view. (B) The target robot's point of view, generated by the observing robot.

paper, inverse model/goal combinations with high applicability were attributed to the target robot as being available actions.

6. Results

Although the extracted 3D depth information was subject to visual noise, leading to some jitter in the constructed visual model, the visual image generated by the inverse visual model was sufficiently accurate to not require any filtering. Figure 6(A) shows the visual scene from the observing robot's point of view, and Figure 6(B) shows the scene generated by the inverse visual model using the internal spatial model.

The observing robot then attempted to take visual perspective by manipulating the internal spatial model according to the viewpoint transformation determined by the visual forward models. Figure 7(A) shows the target robot's view of the scene, and Figure 7(B) shows the result of the observing robot's attempt to take visual perspective.

Figure 8 shows a comparison between the applicability of the inverse models "move to object 1" and "move to object 2" from both the observing robot's perspective (upper graph) and the perspective of the target robot (lower graph). From the point of view of the observing robot, both object 1 (the cuboid object) and object 2 (the cylindrical object) can be seen, and so both the "move to object 1" and "move to object 2" inverse models are applicable. The applicability curves for the two inverse models are very similar, as the objects are positioned close together, and the robot is internally simulating moving to the objects. However, towards the end of the curves, the two inverse models diverge slightly, with the cuboid object being slightly closer to the robot and thus allowing the "move to object 1" inverse model to obtain a higher applicability. The curve in the graphs is due to the exponential form of equation (2).

From the perspective of the target robot (lower graph) only object 1 can be seen, and so only the inverse model "move to object 1" is applicable. Because the object is much closer to the target robot than the observing robot, fewer simulation steps are required to complete the inverse model "move to object 1" and thus it obtains a

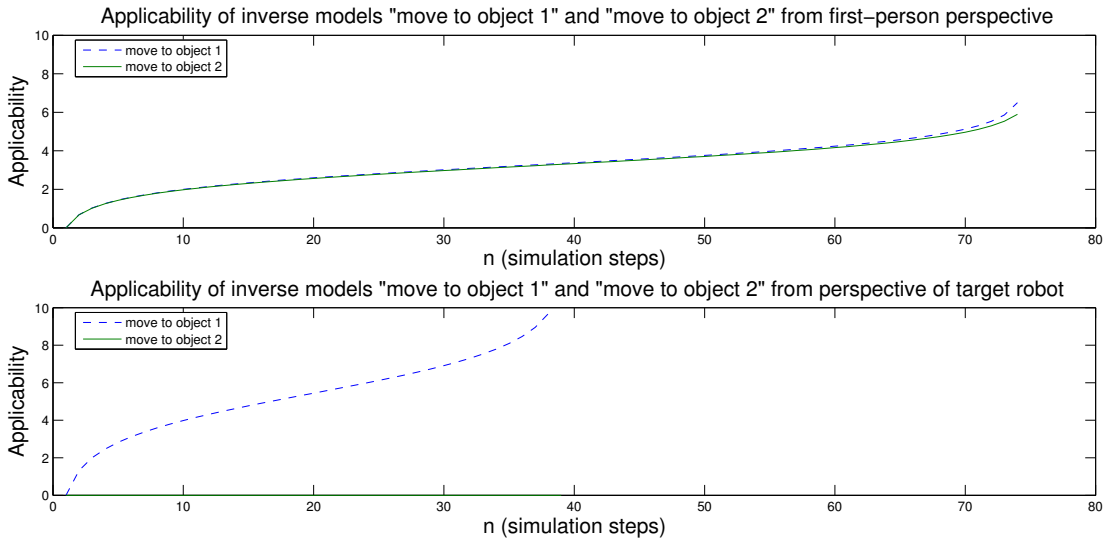

Figure 8: A comparison of inverse model applicabilities generated using the observing robot's first-person perspective (upper graph), or when taking the perspective of the target robot (lower graph). The applicability is re-calculated every time step over the n action simulation steps required to achieve completion. From the observing robot's perspective, both object 1 (the cuboid object) and object 2 (the cylindrical object) can be seen, and so both the "move to object 1" and "move to object 2" inverse models are applicable. From the perspective of the target robot (lower graph) only object 1 can be seen, and so only the inverse model "move to object 1" is applicable.

higher applicability than when used in first-person perspective. The applicability is a relative measure and is not intended to be interpreted absolutely.

7. Discussion

In this paper we took a simulation theory approach to perspective taking, using internal control, visual and spatial models. The fundamental requirement for this approach is the existence of a forward visual model dedicated to extracting a target subject's eye location and gaze direction. This forward visual model acts as part of a larger perspective-taking mechanism that recruits internal models used in first-person perception and action in order to attribute perceptions and actions to the target through simulation.

In the experiment performed in this paper, the observing robot successfully attributed visual perceptions to an observed, "target" robot. This is equivalent to solving the level 2 visual perspective-taking task (Flavell, 1999), which requires subjects to have knowledge that the same scene looks different to other people, who are viewing the scene from different locations.

The experiment performed in this paper went further, with the observing robot attributing potential actions to the target robot using action simulation from the target robot's perspective. However, this attribution is performed using attributed perceptual states derived from observation; it does not take into account any prior knowledge that the target robot may have as regards the location of the objects, and it does not consider the possibility that the target robot may see the cylindrical object as it moves towards the cuboid object. In future work, the observing robot will maintain knowledge about the target robot's *belief states*, allowing for more accurate behaviour prediction.

By attributing visual perceptions to the target robot, the observing robot is also implicitly attributing *visual attention*. A further enhancement to the perspective-taking system will therefore be the addition of an *attention mechanism*, already in development in the BioART lab (Khadhouri and Demiris, 2005), to increase the accuracy of the perspective taking.

Acknowledgments

The authors would like to thank the BioART team - Gavin Simmons, Bassam Khadhouri, Paschalis Veskos and Anthony Dearden, and also Mercedes Lahnstein for her invaluable support and assistance.

References

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

Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel Technol-*

ogy Journal, Q2:1–15.

Dearden, A. and Demiris, Y. (2005). Learning forward models for robotics. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) 2005, to appear.*

Demiris, J. (1999). *Movement imitation mechanisms in robots and humans.* PhD thesis, University of Edinburgh, Scotland, UK.

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

Demiris, Y. and Johnson, M. (2005). Simulation theory for understanding others: A robotics perspective. In *Imitation and Social Learning in Robots, Humans and Animals: Behavioural Social and Communicative Dimensions.* Cambridge University Press.

Flavell, J. H. (1999). Cognitive development: Children's knowledge about the mind. *Annual Review of Psychology*, 50:21–45.

Gallese, V. and Goldman, A. (1998). Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, 2(12):493–501.

Gordon, R. M. (1999). Simulation vs theory-theory. In Wilson, R. A. and Keil, F., (Eds.), *The MIT Encyclopædia of the Cognitive Sciences*, pages 765–766. MIT Press.

Johnson, M. and Demiris, Y. (2005). 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".*

Jordan, M. and Rumelhart, D. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354.

Khadhouri, B. and Demiris, Y. (2005). Compound effects of top-down and bottom-up influences on visual attention during action recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) 2005, to appear.*

Miall, R. C. and Wolpert, D. M. (1996). Forward models for physiological motor control. *Neural Networks*, 9(8):1265–1279.

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

Nichols, S. and Stich, S. P. (2003). *Mindreading.* Oxford University Press.