International Journal of Advanced Robotic Systems, 2(4), pp. 301-308, December 2005.

ARS ADVANCED ROBOTIC SYSTEMS INTERNATIONAL

Perceptual Perspective Taking and Action Recognition

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 in social environments need to be able to recognise and understand the actions of other robots, and humans, in order to facilitate learning through imitation and collaboration. The success of the simulation theory approach to action recognition and imitation relies on the ability to take the perspective of other people, so as to generate simulated actions from their point of view. In this paper, simulation of visual perception is used to recreate the visual egocentric sensory space and egocentric behaviour space of an observed agent, and through this increase the accuracy of action recognition. To demonstrate the approach, experiments are performed with a robot attributing perceptions to and recognising the actions of a second robot.*
Keywords: *perspective taking, action recognition, mobile robotics.*

1. Introduction

One of the most widely researched theories of how we attribute mental states to others is the *simulation theory* (Nichols, S. & Stich, S. P., 2003, Gordon, R. M., 1999). 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.

The HAMMER (Hierarchical Attentive Multiple Models for Execution and Recognition) architecture takes a biologically inspired simulation-theoretical approach to action recognition and imitation (Demiris, Y. & Khadhouri, B., 2005). It achieves this by directly involving the observer's motor system in the action recognition process. During observation of another person'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 recognised 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.

In order to provide meaningful data for comparison, the simulated actions used by the observer during recognition must be generated *as though from the point of view of the other person.* Since the observer's inverse models require first-person data in order to generate actions, this is achieved through *perspective taking*, which represents an egocentric "shift" from the observer to the observed. The data required for the inverse models to operate is therefore derived from consideration of the observed agent's physiospatial circumstances, and not the observer's.

Perspective taking can also be used to increase the accuracy of the action recognition process that is at the heart of the HAMMER architecture. Developed in this article is an approach that uses *perceptual* perspective taking to build up first the *egocentric sensory space*, and then the *egocentric behaviour space* of another person, i.e., the possible goals and actions that are available to that person given his physiospatial circumstances. The egocentric behaviour space can be used to constrain the set of inverse models used for action simulation, reducing the opportunity for matching errors.

2. Background

2.1 Internal Inverse Models

One of the core components of the HAMMER architecture is the *inverse model*. 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, K. S. & Balakrishnan, J., 1997, Wada, Y. & Kawato, M., 1993). In the control litera-

ture, 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 *internal states*, that are used in action execution and recognition:

- 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, Y. & Johnson, M., 2003).

Confidence and *applicability* are scalar metrics, generated through *simulation–comparison processes*, that may take any value. The other states are considered binary states, and are updated during both action generation and simulation.

The HAMMER architecture achieves action simulation by coupling inverse models to *forward models*.

2.2 Internal Forward Models

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 (Wolpert, D. M. & Kawato, M., 1998, Wolpert, D. M. 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.

3. Simulation of Perception

In the HAMMER architecture, current state information for inverse models used in recognition is generated through consideration of the observed agent's physiospatial situation. For inverse models that generate interactions with the environment, this often results in distance metrics, between e.g. end effectors and manipulable objects. This is "perspective taking" inasmuch as perspective taking is using the observed agent as a spatial reference point for geometric calculations.

Through consideration of what the observed agent *perceives* as well as his position, more accurate state information may be generated. For example, objects that the observing agent can see may be occluded or otherwise obstructed from the observed agent's point of view. Considering these objects as potential goals would therefore lead to unnecessary action simulations that would increase the opportunity for error as well as the overall cost of simulation. In addition, objects may present a different aspect to the observed agent than to the observing agent; this effects e.g. angle of approach and grip formation in reaching-and-grasping actions, and so must be taken into account as well as the relative position of the object when using action simulation for action recognition and imitation.

In order to address these issues we develop a system that equips the HAMMER architecture with the capacity for *visual perceptual perspective taking*. In keeping with the simulation theoretical approach, this capacity is achieved through a biologically inspired *simulation of visual perception*. In the same way as action execution and recognition is performed 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 *vision*.

3.1 Internal Vision Models

The generic *forward vision model* is defined as taking two inputs, the first being afferent sensory information in the visual modality, e.g. an image of a visual scene in bitmap format, and the second being the visual parameters with which to process that input, e.g. a colour histogram or shape template for object segmentation. The output from the model is the generated abstract state resulting from processing, e.g. geometric co-ordinates in image- or world-space. In the architecture developed here, forward vision models feed state information into the coupled inverse and forward models of the HAMMER architecture. It is important to note that the "forward" in the forward vision model

described here does not entail a temporal predictive ability, as used in (Demiris, Y. & Johnson, M., 2003, Wolpert, D. M. et al., 2003), but rather describes the feed-forward nature of the information flow through the model of the visual system.

An example of a forward vision model would be one that takes stereoscopic visual information from binocular cameras, visual object descriptions as parameters, and generates as output a three-dimensional spatial representation of the objects' locations in an egocentric frame of reference. This forward vision model may in turn be composed of forward vision models performing edge detection and background subtraction.

Similarly, the *inverse vision model* is defined as having two inputs and one output. The inverse vision model takes as input visual object properties retrieved from visual memory (e.g. colour, shape, etc), and their desired state (e.g. relative positions, depth, occlusions, etc.), and produces as output the visual image that results from reconstructing these inputs, in the same pictorial format as the visual afference supplied to the forward vision model. Analysis and transformation of the image constructed by the inverse vision model may then proceed through use of the forward vision models, thus involving them in a process of *perception simulation* (Johnson, M. & Demiris, Y., 2005).

In the same way as inverse and forward models interact during the action execution and recognition process, through predictive control and simulation, so to do the forward and inverse vision models during construction of visual perception and perspective taking. While forward vision models perform the forward process of converting visual afference into internal states, the inverse vision models are used in the perception process to enhance incomplete visual input, and perform shape and template matching (Kosslyn, S. M., 1994).

3.2 Perspective Taking

Internal vision models for achieving perspective taking can now be defined. For the perspective taking approach currently used in the HAMMER architecture, we define a forward vision model that segments end-effectors and objects in the visual scene and computes both their locations in world coordinates, and the distances between them. This forward vision model, and the spatial location states it produces, is also used as part of the perceptual perspective taking process developed below.

Another forward vision model, crucial for perceptual perspective taking, is one that processes the *gaze direction* of the observed agent. This forward vision model identifies the visual sensors of the observed agent and calculates their locations in world coordinates, as well as the direction vectors corresponding to direction of gaze. These vectors, taken together with the object coordinates extracted by the forward vision model described in the previous paragraph, form a geometric transform that may be used to take the *spatial* perspective of the observed agent. By involving these forward vision models in a perception simulation process with the *inverse* vision models, spatial perspective taking may be extended to re-create the visual egocentric sensory space of the observed agent.

Inverse vision models, though integral to first-person perception construction (in the same way as the forward models in HAMMER are integral to action execution), are therefore developed here for the purpose of reconstructing the observed agent's egocentric sensory space during perspective taking. In the visual modality, the egocentric sensory space may take the form of the visual afference direct from the visual sensors, or it may be equally well formed at some later stage of visual processing, for example a stereovision disparity map, figure-ground separation or object segmentation. In this paper we develop the latter approach, using the inverse visual models to reconstruct scenes of segmented objects separated from ground.

Visual perceptual perspective taking therefore proceeds, according to the simulation theory approach, through the following stages. Forward vision models for extracting object state and determining gaze direction work on data from the visual sensors in order to produce the geometric transforms described above, resulting in spatial perspective taking. These transformed spatial data are fed back into the inverse vision models as desired states, along with visual descriptions corresponding to the objects retrieved from visual memory. The output from the inverse vision models is a re-creation of the observed agent's visual egocentric sensory space, including occlusions and other visual cues. As mentioned above, this need not be a full recreation of the observed agent's visual sensor output, but can be at some later stage of visual processing, and thus may be an abstraction of the observed agent's visual sensing. To complete the simulation of the observed agent's perception, the re-created egocentric sensory space is then processed by the forward vision models that feed the inverse models in the HAMMER architecture, thereby presenting the third-person state information necessary for action recognition to the inverse models in the first-person format they require.

The inverse models use the state data derived from the egocentric sensory space to determine the *egocentric behaviour space*. The egocentric behaviour space is defined here as being the set of inverse models that may be executed given an agent's instantaneous physiospatial circumstances and the perceptual information available to him. The egocentric behaviour space is therefore built up from two inverse model states defined in section 2.1, the *eligibility* and the *applicability*

level. The calculation of eligibility and applicability level is described in section 5.3.

4. Experiments

The forward and inverse vision models were implemented on an ActivMedia Peoplebot, along with an implementation of the HAMMER architecture, 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 first set of experiments was for the Peoplebot to re-create the egocentric visual sensory space and egocentric behaviour space of the target, by taking its perspective, and to compare this with the egocentric visual sensory space and egocentric behaviour space derived from the observing robot's first-person perspective. The second set of experiments was to feed the implemented HAMMER architecture with the egocentric sensory and behaviour spaces, and assess the subsequent action recognition performance as the target robot performed move-to-grasp-object actions.

4.1 Experimental Setup

Fig. 1 shows a plan view of the experimental setup. 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. The objects were placed on the table in such a way that due to the construction of the Peoplebots, the observing robot would be unable to grasp either object using its inverse models. In the same manner, only the cuboid block was manipulable by the observed, "target" Peoplebot. Fig. 2 shows a PeopleBot picking up an object.

5. Implementation

5.1 Forward Visual Models

Forward visual models were implemented using the ARToolkit (Billinghurst, M. et al., 2001). 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 as fiducials to the table, objects, and target robot. Since the ARToolkit can also extract orientation information, a fiducial was also attached to the target Peoplebot's camera, in order to extract its gaze direction.

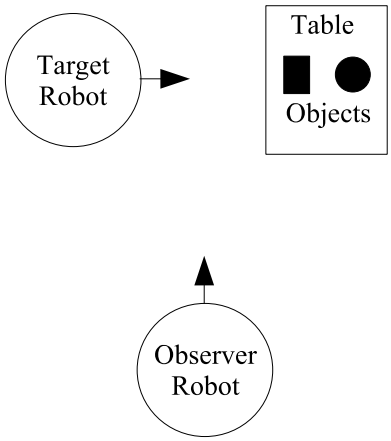

Fig. 1: Plan view of the experimental setup. The arrows indicate robot camera direction. The cuboid and cylindrical objects are placed on the table such that the observer robot can see both but grasp neither, and the target robot can see and grasp only the cuboid object.

Fig. 2: A PeopleBot moves in to grasp an object. The Peoplebot's grippers do not extend, so it can only grasp objects at the very edge of a table.

5.2 Inverse Visual Models

To construct visual scenes from visual object descriptions and locations, the inverse visual models used the OpenGL graphics library (www.opengl.org). 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 those of the observing robot's camera, a Canon VCC4.

5.3 HAMMER Architecture

To clearly demonstrate the contribution of the perspective taking to action recognition, a version of the HAMMER architecture was implemented without hierarchies or attention. The architecture was equipped with ten inverse models: "move to grasp object 1" (the cuboid) at speeds of 10, 20, 30,

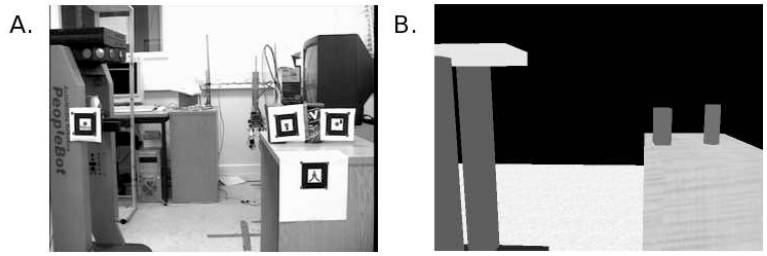

Fig. 3: (A) The visual scene from the observing robot's point of view. (B) The internally generated image of the observing robot's point of view.

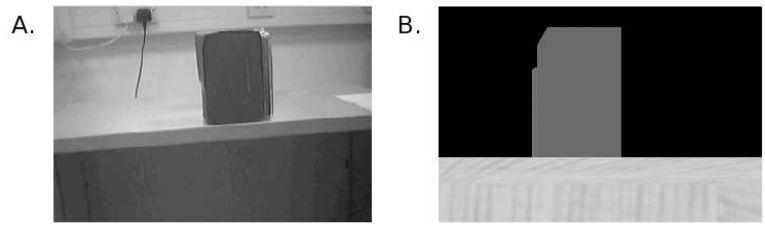

Fig. 4: (A) The visual scene from the target robot's point of view. (B) The target robot's point of view, generated by the observing robot.

40, and 50 mm/sec, and "move to grasp object 2" (the cylinder) at the same distribution of speeds. A generic "move to grasp object" inverse model could have been implemented with the goal object and movement speed as parameters, but this would not have effected the results. The forward model used was a kinematics model of the Peoplebot, which generated one time-step predictions of position through numerical integration of velocity (as in (Demiris, Y. & Johnson, M., 2003)).

The implemented HAMMER architecture was fed with a current state vector S_t produced by the forward vision models. The state vector comprised the x, y, z positions of each fiducial. The inverse models in the architecture 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 state vector λ:

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

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 0.05.

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 not within this set ($S_t \in \Upsilon$), then the inverse model was eligible and became part of the egocentric behaviour space.

During the first set of experiments, 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* of each inverse model A_t was then accumulated for the n^{th} simulation iteration according to:

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

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. In the experiments, 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 $N = 300$. The resulting applicability level determined how useful each inverse model was for achieving its goal. Inverse models with

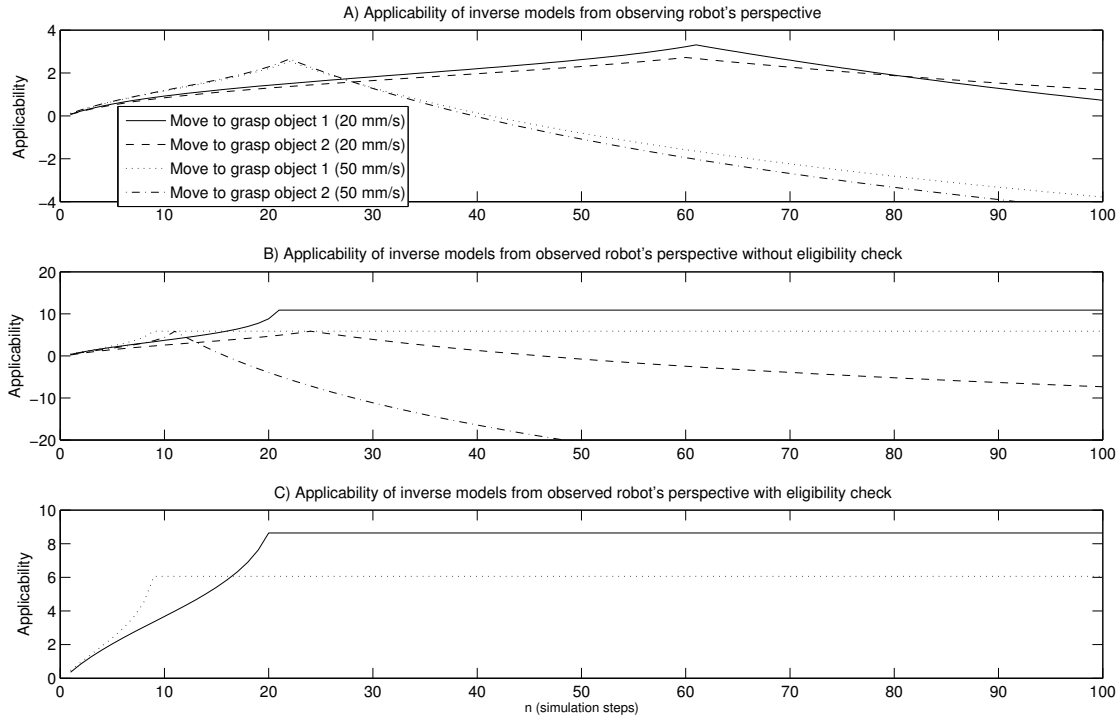

Fig. 5: Applicability levels of inverse models during construction of egocentric behaviour space. Four out of the ten inverse models are shown for clarity. (A) Observing robot's perspective; (B) Observed robot's perspective, no eligibility check; (C) Observed robot's perspective with eligibility check.

applicability greater than zero were attributed to the target robot as being part of its egocentric behaviour space.

During action recognition in the second set of experiments, the forward models in the HAMMER architecture produced a prediction $\hat{S}_t$ as to the result of the motor command generated by the coupled inverse model, and this was compared with the actual resulting state, S_t. The resulting prediction error P_e was then used to calculate the *confidence* level of the inverse model. In the implementation of the HAMMER architecture used here, the prediction error was calculated as being the sum over the M state elements, of the absolute difference between the predicted state and the actual state:

$$P_e = \sum_{i=1}^{M} \left| S_{t,i} - \hat{S}_{t,i} \right| \tag{3}$$

At each timestep during recognition, the inverse models had their confidence C_t updated as follows:

$$C_t = \begin{cases} C_{t-1} + \dfrac{1}{\epsilon_2} & \text{if } P_e < \epsilon_2 \\ \\ C_{t-1} + \dfrac{1}{P_e} & \text{otherwise} \end{cases} \tag{4}$$

In a winner-take-all approach, the inverse model with the highest level of confidence at the end of the demonstration was selected as being the inverse model that matched best with the observed action.

Initial confidences were zero for all inverse models. In the following experiments, ϵ_2 was chosen to be 0.001.

6. Results

The results from the first set of experiments are shown in figures 3, 4, and 5. Fig. 3(A) shows the visual scene from the observing robot's point of view. Fig. 3(B) shows the result of visual processing in constructing the observing robot's egocentric sensory space.

The observing robot then attempted to take the perspective of the target robot using the transforms determined by the visual forward models. Fig. 4(A) shows the target robot's view of the scene, and Fig. 4(B) shows the result of the observing robot's reconstruction of the target robot's egocentric sensory space.

The graphs in Fig. 5 show the results of the applicability calculation during the construction of both the observing and observed robots' egocentric behaviour spaces. Fig. 5(A) shows the applicability levels of four inverse models when simulating from the observ-

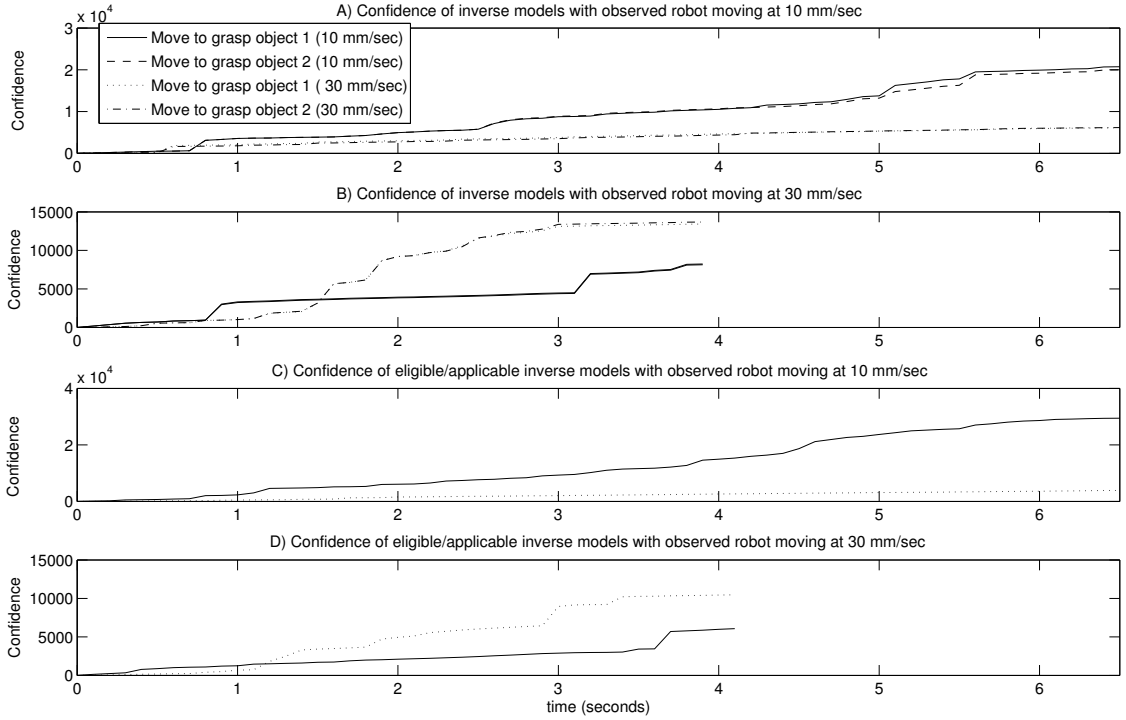

Fig. 6: Confidence levels of inverse models during action recognition. Four out of the ten inverse models are shown for clarity. (A) 10 mm/sec movement with no perspective taking; (B) 30 mm/sec movement with no perspective taking; (C) 10 mm/sec movement with perspective taking; (D) 30 mm/sec movement with perspective taking.

ing robot's point of view. The applicability level of all inverse models increases before reaching a peak and thereafter decreasing. The inverse models for 50 mm/sec peak before the inverse models for 20 mm/sec. Fig. 5(B) shows the applicability level of the same inverse models but from the target robot's point of view. In this experiment, the eligibility of the inverse models was not used in the construction of the egocentric behaviour space before simulation. The inverse models for moving to grasp the cuboid object (object 1) achieve a sustained high applicability level, whereas the applicability level of the inverse models for moving to grasp the cylindrical object (object 2) fall below zero. Fig. 5(C) shows the applicability levels of inverse models from the target robot's point of view, with the eligibility of the inverse models first determined by taking the target robot's perspective and reconstructing its egocentric sensory space. The inverse models for moving to grasp the cuboid object are included in the egocentric behaviour space, and both achieve a positive applicability level, as in Fig. 5(B).

The graphs in Fig. 6 show the confidence level of inverse models during recognition of the robot moving to grasp the cuboid object (object 1). Fig. 6(A) shows the confidence levels of four inverse models, move to grasp object 1 at 10 and 30 mm/sec, and move to grasp object 2 at 10 and 30 mm/sec. With the target robot set to move to grasp the cuboid object at 10 mm/sec, the graph clearly shows the inverse models for moving at 10 mm/sec with a higher confidence level than those for 30 mm/sec. Fig. 6(B) shows the same experiment but with the target robot set to move to grasp the cuboid object at 30 mm/sec. In this experiment, not only is the experimental time shortened (since the robot reaches the object faster), but the inverse models for moving at 30 mm/sec achieve a higher confidence level than those for 10 mm/sec. Figures 6(C) and (D) show the same experiments but with the initial eligibility check and applicability level calculation resulting from perspective taking. Inverse models for grasping the cylindrical object are dropped from the observed robot's egocentric behaviour space, and only the inverse models for grasping the cuboid object are used in recognition.

7. Discussion

Although the extracted 3D depth information was subject to visual noise, the construction of the observing robot's egocentric sensory space (Fig. 3) was sufficiently accurate to not require any filtering. The difference between the observed robot's actual point

of view and the observing robot's re-creation of the observed robot's egocentric sensory space (shown in Fig. 4) is due to slight inaccuracies in the extracted angles of gaze direction and camera position, but is sufficiently accurate for construction of the observed robot's egocentric behaviour space.

As described in section 4.1, neither of the objects is directly graspable by the observing robot. It is for this reason that all of the inverse models shown in Fig. 5(A) end up with decreasing applicability levels. Initially, the applicability levels are increasing; the peak arrives when the robot hits the table (in simulation) and becomes "stuck". Thereafter, the robot makes no progress towards picking up the objects, and the applicability levels decrease. The peak for the 50 mm/sec inverse models occurs before that of the 20 mm/sec inverse models, since the robot hits the table sooner when moving at that speed. Fig. 5(B) shows the same experiment but from the observed robot's point of view. The observed robot is able to directly grasp object 1, and so the inverse models for grasping that object maintain a high applicability level. The inverse models for grasping the second object respond in the same way as in the experiment with the observing robot. Fig. 5(C) shows the result of perspective taking on the applicability calculation. Since the cylindrical object (object 2) cannot be seen from the observed robot's point of view, it is rendered ineligible and dropped from the egocentric behaviour space. The inverse model for moving to grasp the object at 50 mm/sec achieves a lower applicability level than the inverse model for 20 mm/sec, even though it reaches the object sooner. Since the applicability calculation of equation 2 rewards accuracy in achieving the goal state as well as discounting over time, this may be due to inaccuracy in reaching the goal as compared to the slower inverse model.

The graphs of figures 6(A) and (B) demonstrate the HAMMER architecture's capacity for recognising not only an action but also the speed at which it is performed. The confidence levels of the 10 mm/sec inverse models in Fig. 6(B) are higher than those of the 30 mm/sec inverse models towards the beginning of the experiment, because the robot accelerates through 10 mm/sec to reach the demonstration speed of 30 mm/sec. However, at this higher demonstration speed, the observing robot *mis-recognises* the action, since the inverse model for grasping object 2 achieves the highest confidence. By taking the perspective of the observed robot and constructing its egocentric behaviour space this mis-recognition is avoided. Inverse models for grasping object 2 are neither eligible nor applicable, and so are not included in the egocentric behaviour space used for action recognition. Figures 6(C) and (D) show the subsequent improvement in action recognition.

8. Conclusions

The HAMMER architecture provides an efficient framework for accurately recognising actions. However, at higher action speeds, where the visual information-to-visual noise ratio is low, actions may be mis-recognised and action goals be mis-attributed to unobtainable objects. By using perceptual perspective taking to calculate the eligibility and applicability of inverse models, the egocentric behaviour space can be constructed and used to increase the accuracy of action recognition.

References

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

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

Demiris, Y. & Khadhouri, B. (2005). Hierarchical attentive multiple models for execution and recognition. *Robotics and Autonomous Systems, to appear.*

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

Johnson, M. & Demiris, Y. (2005). Perspective taking through simulation. In *Proceedings of TAROS*, pages 119–127.

Kosslyn, S. M. (1994). *Image and Brain: The resolution of the imagery debate.* MIT Press.

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

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

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

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

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