BioSystems 128 (2015) 37–47

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

Inference of other's internal neural models from active observation

Kyung-Joong Kim [a], Sung-Bae Cho [b,*]

[a] Department of Computer Science and Engineering, Sejong University, Seoul, South Korea
[b] Department of Computer Science, Yonsei University, Seoul, South Korea

ARTICLE INFO

Article history:
Received 13 October 2014
Received in revised form 6 January 2015
Accepted 19 January 2015
Available online 21 January 2015

Keywords:
Theory of mind
Robot
Physics-based simulation
Active learning
Neural network
Evolutionary computation
Estimation–exploration algorithm

ABSTRACT

Recently, there have been several attempts to replicate theory of mind, which explains how humans infer the mental states of other people using multiple sensory input, with artificial systems. One example of this is a robot that observes the behavior of other artificial systems and infers their internal models, mapping sensory inputs to the actuator's control signals. In this paper, we present the internal model as an artificial neural network, similar to biological systems. During inference, an observer can use an active incremental learning algorithm to guess an actor's internal neural model. This could significantly reduce the effort needed to guess other people's internal models. We apply an algorithm to the actor–observer robot scenarios with/without prior knowledge of the internal models. To validate our approach, we use a physics-based simulator with virtual robots. A series of experiments reveal that the observer robot can construct an "other's self-model", validating the possibility that a neural-based approach can be used as a platform for learning cognitive functions.

1. Introduction

Robots can represent a simplified model of human behavior, whereby the robot senses its environment and reacts to various input signals. The robot's 'brain' controls its body in response to the input signals using artificial neural networks. The topology and weights of the neural network characterize the behavioral properties of the robot. Recently, several investigations have used robots in order to gain insight into human cognition by creating a simplified analogous problem (Bongard et al., 2006; Webb, 2001; Floreano and Keller, 2010). Bongard et al. built a starfish robot; however, it was unaware of its own body shape (Bongard et al., 2006). Using an estimation–exploration algorithm (EEA) (Bongard and Lipson, 2007), the robot was able to successfully create a self-model of its body shape using an iterative estimation and exploration procedure. In the estimation step, the robot searched multiple candidates to determine its body shape. Subsequently, in the exploration step, the algorithm determined the actions that most strongly agreed with the multiple candidate body shapes.

Unlike self-modeling, however, theory of mind (ToM) is a high-level cognitive function that models the mental states (beliefs, intents, desires, imagination, knowledge, *etc.*) of another entity. In robotic studies, robots have demonstrated the ability to mimic the behavior of humans or to decode the intentions of a third party (both human and robot). For example, Scassellati implemented Baron-Cohen's ToM model for the humanoid robot COG (Scassellati, 2002). Breazeal et al. demonstrated that an animal-like robot could pass the false-belief test widely used to test ToM in young children (Breazeal et al., 2005). Furthermore, Buchsbaum et al. carried out simulations in which one agent attempted to determine another agent's behavior using rat-like characters (Buchsbaum et al., 2005). In this particular study, the observer exploited his own behavior tree to infer others' intentions.

However, few reports have described the representation of another entity's mind as a neural circuit. Revealing an internal neural model based on observations is a challenging task. However, there is great potential for using neural networks as internal models, because it would mimic the underlying mechanisms of human representations in the form of neural connections. Many different definitions of the self and other's self-representations exist, ranging from symbolic states to complex neural models. For example, Bongard et al. (Bongard et al., 2006) used the morphological structure of a robot as a self-model. The robot had no physical model of itself on which to base an understanding, and attempted to construct models of its body using iterative estimation–exploration steps. Kim and Lipson used a simple feed-forward network to represent the minds of other (Kim and Lipson, 2009a,b,b).

In this paper, we propose the use of active incremental learning to infer the internal neural models of other entities both with and without prior knowledge (Fig. 1). We used two robots, referred to

* Corresponding author.
E-mail addresses: kimkj@sejong.ac.kr (K.-J. Kim), sbcho@cs.yonsei.ac.kr (S.-B. Cho).

http://dx.doi.org/10.1016/j.biosystems.2015.01.005
0303-2647/

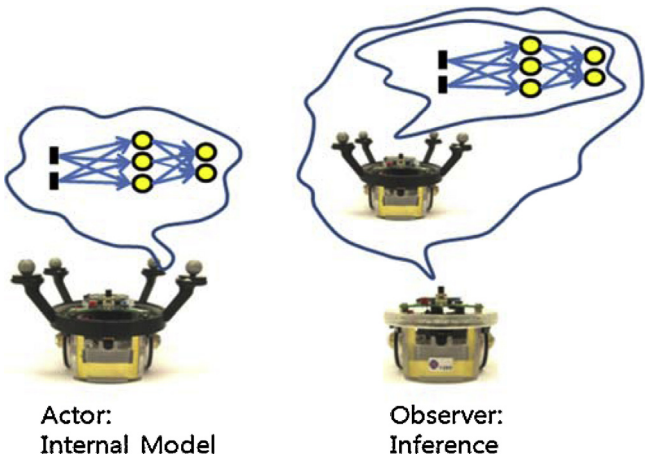


Fig. 1. Inference of other's internal models.

as the actor and the observer. The actor used a neural controller (implemented as an artificial neural network) to control its behavior based on sensory information. The observer monitored the behaviors of the actor and attempted to infer the actor's internal model from these observations. The observer used the inferred self-model of the actor to predict the actor's future behavior. In this approach, instead of programming the other's internal model manually, the observer attempted to predict the other's self-model interactively. The observer robot started from a single actor trajectory and invited the actor robot to demonstrate additional trajectories, which were then used to infer information about the actor's self-model using the EEA method (Bongard and Lipson, 2007).

In particular, we tested the impact that prior knowledge had on the actor's internal model. Initially, we assumed that the actor and observer were the same species and that the observer could use his self-model (neural topology). Therefore, the ToM problem is formulated as the inference of the connection weights given the shared structure. We subsequently assumed that the two robots are different species and that the actor could not use his self-model for the ToM. As a result, the observer needs to search for the architecture of the neural network and the weights simultaneously to infer the other's self-model. We used a physics-based simulation to run the ToM experiments, which show the potential of this approach given the two experimental conditions.

The rest of this paper is organized as follows. In Section 2 we describe related research, including the research on ToM in robots. In Section 3 we apply the estimation–exploration algorithm for the robotic ToM. Finally, in Section 4, we present our experimental results.

2. Background

2.1. Inference of other's mind in humans

ToM is the ability to attribute mental states to oneself and others, and to understand that others have different beliefs, desires, and intentions from one's own (Premack and Woodruff, 1978). The first paper on ToM, published in 1978 by Premack and Woodruff, posed the question, "Does the chimpanzee have a theory of mind"? Since then, many articles on ToM in human and non-human primates have been published (Call and Tomasello, 2008). Attempts have been made to reveal the existence of ToM in many species, including monkeys, and elephants, and ToM has been used to inform studies related to fundamental mechanisms and how certain conditions, such as autism, may develop (Baron-Cohen, 1995). Recently, brain imaging technology has been used to demonstrate the activation of specific areas in the brain associated with ToM (Siegal and Varley, 2002).

In the 30 years since the introduction of ToM, researchers have proven that chimpanzees have ToM, but that they cannot understand each other to the degree that humans do (Call and Tomasello, 2008). Herrmann et al. compared ToM ability among humans, chimpanzees, and orangutans using gaze-following and intention-understanding tasks (Herrmann et al., 2007) showing that humans outperformed chimpanzees and orangutans. In humans, ToM has been shown to be related to neural development disorders that are characterized by impaired social interaction and communication; for example, childhood autism may be associated with a deficit in ToM (Baron-Cohen, 1995). Baron-Cohen compared normal subjects and subjects with autism and Down syndrome using a belief question to test ToM, finding that subjects with Down syndrome were similar to the control group; however, 80% of autistic children failed to show ToM (Baron-Cohen, 1995).

How the ToM works is not well understood. There are several theories to explain these high-level cognitive functions. Several robotics researchers have used robots in attempts to better understand these theories (Scassellati, 2002; Breazeal et al., 2005). However, debate among neuroscientists concerning the evidence supporting these different hypotheses persists (Siegal and Varley, 2002; Saxe, 2009). ToM theories can be classified into three categories: modular, theory–theory, and executive function theories (Youmans, 2004).

- In the modular view (supported by Baron-Cohen, 1995), ToM is functionally dissociable from other cognitive functions, and it is assumed that there is one or more neural structures specifically dedicated to this function. Baron-Cohen assumed that the ToM process includes an intentionality detector, an eye-direction detector, a shared-attention mechanism, and a ToM mechanism (Baron-Cohen, 1995).
- According to the theory–theory school, a child has a theory about how other minds operate, which evolves over time and with experience.
- Some theorists argue that a distinct ToM does not exist and that executive functions are sufficient to explain the skills involved in ToM (Ozonoff et al., 1991).

We believe that developing and testing ToM models using robots may provide insight into some of these complex questions.

2.2. Computational approaches for ToM

Robots are increasingly being used as a platform to test theories of human behavior and cognition (Webb, 2001). In a broad sense, a robot includes virtual agents, characters, simulated robots, and real robots. Recently, interesting interdisciplinary research has shown the effectiveness of a robot-based approach. For example, Wischmann et al. investigated the emergence of communications using physics-based robot simulators (Wischmann et al., 2012). Bongard et al. also used robots and a physics-based robot simulator, based on the open dynamics engine, to demonstrate robot self-modeling (Bongard et al., 2006).

Because ToM is an important cognitive function in humans, researchers have applied the concept to virtual agents, virtual characters, simulated robots, and real robots (see Table 1 for further details). In most studies, more than two robots were used, and each assumed the role of either "actor" or "observer". The observer robots inferred the internal model of the actor robot from observations of the actor's behavior. If the observer was successful in revealing the internal model of the actor, the model used in the estimation could be used to predict subsequent behavior by the actor. Because this inference is a kind of reverse-engineering task,

Table 1
Summary of ToM research for artificial agents (robots, virtual characters, *etc.*).

	Reference	Observer	Actor	Other's self-representation	Modeling methods	Specific tasks
Simulation study	(Peters, 2006)	Virtual character	Virtual character	–	Symbolic memory	Conversation
	(Kaliouby and Robinson, 2004)	Computer	Human (video)	Six discrete mental states	Bayesian networks with facial expression and head gestures	Mind-reading dataset for individuals with autism
	(Buchsbaum et al., 2005)	Virtual character	Virtual character	–	Action recognition based on simulation theory	–
	(Bosse et al., 2007)	Virtual agent	Virtual agent	BDI model	Belief–desire–intention (BDI) modeling	Employer's task-avoidance scenario
	(Pynadath and Marsella, 2005)	Virtual agent	Virtual agent	Three discrete mental states	Nested belief modeling on agent-based simulation	School violence scenario
	(Takano and Arita, 2006)	Virtual agent	Virtual agent	–	Predicting other's velocity vector	Collision avoidance
	(Zanlungo, 2007)	Virtual agent	Virtual agent	–	Predicting other's velocity vector	Collision avoidance
	(Kondo and Nishikawa, 2003)	Virtual agent	Virtual agent	Eight discrete actions	Predicting other's discrete action by a neural network	Carrying a stick
	(Bringsjord et al., 2008)	Virtual character	Avatar controlled by human	Discrete	Logical inference based on a ToM statement	False-belief test
Physical implementation	(Kelley et al., 2008)	Mobile robot	Human	Three activities (following, meeting, and passing by)	Hidden Markov model	Understanding intent
	(Breazeal et al., 2005)	Physical robot	Human	–	Goal inference based on simulation theory	False-belief test
	(Scassellati, 2002)	Humanoid	Human	Two discrete intentions (attraction and repulsion)	Implementation based on Leslie's model and Baron-Cohen's model of ToM	Interaction with toys
	(Yokoya et al., 2007)	Humanoid	Human	Hierarchical neural network	Extension of self-model (recurrent neural network)	Physical object moving
	(Demiris and Johnson, 2003)	Mobile robot	Mobile robot	Five discrete behaviors	Inverse–forward models	Gripper movement
	(Takanashi et al., 2007)	Mobile robot	Mobile robot	Five discrete behaviors	Action recognition based on simulation theory	Robot soccer

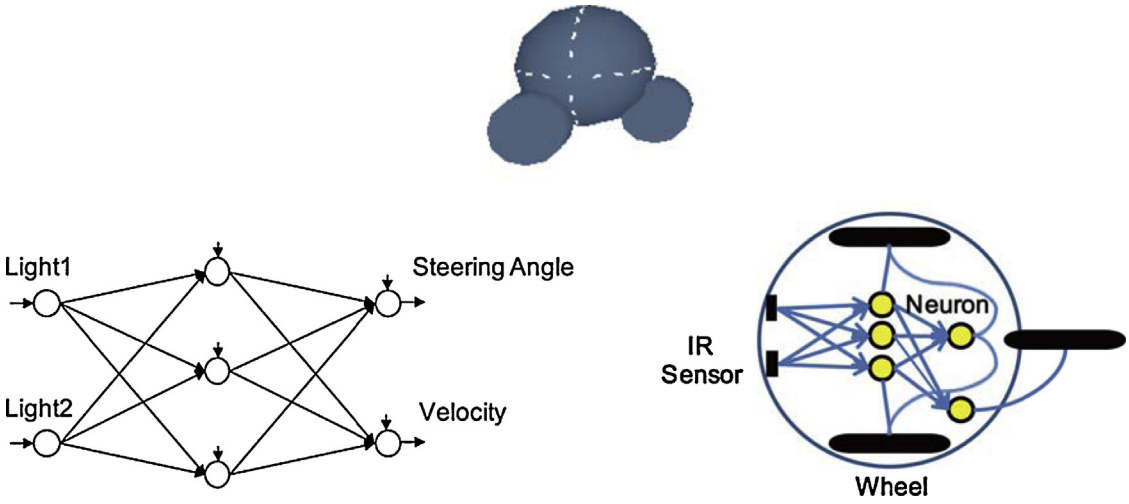


Fig. 2. A virtual robot and the neural controller.

it is not a straightforward problem for robots. In previous research, the actor's self-models have been intentionally simple, typically allowing for only a few discrete states.

3. Proposed method

In this paper, we propose the use of active incremental learning to infer an observer's internal model. There are two robots in this environment; one robot is an actor and the other is an observer. The actor controls itself using a feed-forward neural network (NN), with the inputs to the NN being sensory information, and the outputs control the speed and direction of the robot. The observer monitors the behavior of the actor and attempts to infer its internal neural model with/without prior knowledge. In the observation, the observer continuously estimates the models and explores the new data required to accelerate the learning. The model is successful if the observer can predict the behavior of the actor. The process consists of three steps:

- Step 1: actor learning
  - One robot (*i.e.*, the actor) learns to move toward a light source (*i.e.*, food source) by evolving an "innate" NN. This is not necessarily an efficient motion; it can be any arbitrary complex behavior.
- Step 2: observer learning
  - The observer can access the positions of the actor and the light source. It is presumed that the light sensory inputs to the actor are not accessible to the observer.
- The other robot (the observer) observes the actor's trajectory and uses the path to reverse engineer the actor's innate NN. Additional paths help the observer to refine or refute models of the actor's NN.
- The observer determines where to attract the actor so as to better expose the actor's NN, in order to refine or refute various models of the actor's NN. It is possible to use the light source (*i.e.*, food source) to attract the actor.
- If the accuracy of the predictions is poor, then Step 2 is repeated.
- Step 3: actor exploitation
  - The observer determines where to place the light source in order to elicit the desired behavior from the actor (*e.g.*, making the actor reach a specific target location).

A neural controller was embedded in the robot to enable it to process inputs from the environment, so that it can control the speed and direction of the wheels. The neural controller of the actor was not programmed; instead, it evolved from a trial-and-error process based on its environment. The environment had one light source, representing the robot's goal. The purpose of ToM is not to simply imitate the specific behaviors of the actor, but to estimate the actor's internal neural models. Although it is not possible to reconstruct the exact, original neural network from observation alone, ToM can provide alternatives to the original models. This is also true for humans, where effective self-models of others may differ from one observer to another.

3.1. Actor learning

We performed our experiments using simple wheeled robots controlled with the PhysX engine[1] (physics-based simulator) (Fig. 2). Each robot had three wheels and two sensors that detected the light levels around the robot. The robot used this information to control its wheel velocity. The front wheel controlled the robot's direction, and the two rear wheels controlled the speed. An innate neural network was used to process the sensory inputs and to generate outputs (wheel speed and steering angle). We used 17 connection weights, including bias.

We used an artificial evolution algorithm to train the actor's behavior (Floreano and Keller, 2010; Nolfi and Floreano, 2004). This behavior may not necessarily be efficient in terms of its motion and it can be arbitrarily complex behavior. In the learning environment, the goal light source was placed at (*L_X*,*L_Y*), and the robot was initially placed at (0, 0). The fitness function was defined as follows:

$$f = \frac{1}{\sqrt{\sum_{t=1}^{\mathrm{MAX_{STEP}}} (x_t - L_X)^2 + (y_t - L_Y)^2} + 1}$$

where (x_t, y_t) was the position of the robot at step t. For each step, the robot processed the input signals to actuators using its neural network and the simulator calculated the status of the robot with the motor outputs after a pre-defined amount of time (Δt).

A self-adaptive evolution strategy (ES) was used to evolve the weights of the neural network. This ES has been successfully used in engineering applications, including analog circuit design and racecar controller optimization (Kim and Cho, 2012; Kim et al.,

[1] http://www.geforce.com/hardware/technology/physx.

2012). Initially, P neural networks were generated randomly. The weights (including bias weights) were generated from a uniform distribution over $[-0.2, 0.2]$. Each weight had a corresponding self-adaptive parameter, initialized to 0.05. The mutation operator is

$$\sigma'_i(j) = \sigma_i(j)\exp(\tau N_j(0,1)),$$
$$w'_i(j) = w_i(j) + \sigma'_i(j)N_j(0,1)$$

where N_w is the number of weights, $\tau = 1/\sqrt{2\sqrt{N_W}}$, $\sigma_i(j)$ are the self-adaptive parameters, and $w_i(j)$ is the j^{th} weight of the i^{th} neural network. $N_j(0, 1)$ is a standard Gaussian random variable, which was re-sampled for every j.

Each neural network generates one offspring using the mutation operator, yielding $2 \times P$ neural networks (parents plus offspring). The fitness of each NN was evaluated based on the fitness function. We allowed only the fittest P NNs to survive to the next generation.

3.2. Observer learning

The task of the observer robot is to observe the path of the actor robot and then infer the actor's innate neural controller using the reverse-engineering algorithm. Because the actor's neural controller has evolved, rather than having been programmed manually, the behavior of the robot is not trivial. It exhibits differing behavioral patterns depending on the starting position. To facilitate observer learning, it is important that the observer robot correctly observe the trajectories of the actor. However, it is not feasible to observe all of the trajectories of the actor at all possible starting positions. Instead, the observer actively suggests the starting position of the actor so as to better expose the actor's NN, in order to refine or refute models of the actor's NN.

The observer repeats the estimation (searching for NNs' that show similar behaviors to the observed paths) and exploration steps (ranking the next starting position of the actor using the NNs' found so far) to infer the innate actor's NN. Initially, the center of the environment is set as the actor's starting point. Using the first trajectory, the observer runs an estimation algorithm multiple times (N times with different random seeds). The estimation step produces N candidate NNs' (one from each run). The fitness function is based on the similarity between the actor's trajectory and the simulated trajectory for a given estimated NN.

At this point, the observer has multiple candidates for the innate actor's NN. In other words, the observer has multiple hypotheses about the actor's innate NN. Using them, it is possible to predict the trajectories of the actor for all possible starting positions. For some points, their predictions might be similar. However, in some cases, the predictions might be very different. If it is possible to get one more trajectory from the actor, then it is desirable to request the point at which the predictions disagree (Freund et al., 1997). This process determines the new condition (*i.e.*, the starting positions of the actor) that induces the maximal disagreement among the predictions derived from the candidate models.

One cycle includes both the "estimation" and "exploration". In the subsequent cycle, the observer reuses the candidate models that were learned from the previous cycle to initialize the population in the evolutionary search (incremental evolution). It simply copies the evolutionary search's last population ("estimation") from the previous cycle, rather than initializing it randomly. The cycles continue until a convergence criterion is met. Fig. 3 shows an overview of the estimation and exploration learning algorithm (EEA). The initial population of NNs in the estimation step is copied from the last exploration step. In each cycle, one trajectory is added to the estimation bank (active sampling), and the EEA returns the N best matches to the actor's internal neural model.

The difficulty for the observer is dependent on their prior knowledge of the actor's innate NN. In this research, we restricted the observer's prior knowledge for the different levels. In all of the cases, the observer could only see the position (x–y coordinates) of the actor at each step. The angle of the actor robot was not used. In the first scenario, we used a self-adaptive evolutionary search to infer the connection weights. Because the observer has information on the other robot's brain structure from his prior knowledge, the optimization searches for the connection weights. For the search, the self-adaptive evolutionary search is adopted.

For the second settings, the evolutionary search should optimize both the topology and the connection weights. Beginning with a simple neural network, it continuously applies a set of mutation operators to create new neural network. The following shows evolutionary algorithms used to search without prior knowledge.

- Step 1: initialize the population with a random neural network, which has two input neurons and two output neurons (based on observation). For each NN, one hidden neuron is created randomly (its bias is chosen randomly) with two connection links (random weights). Each link connects the hidden neuron

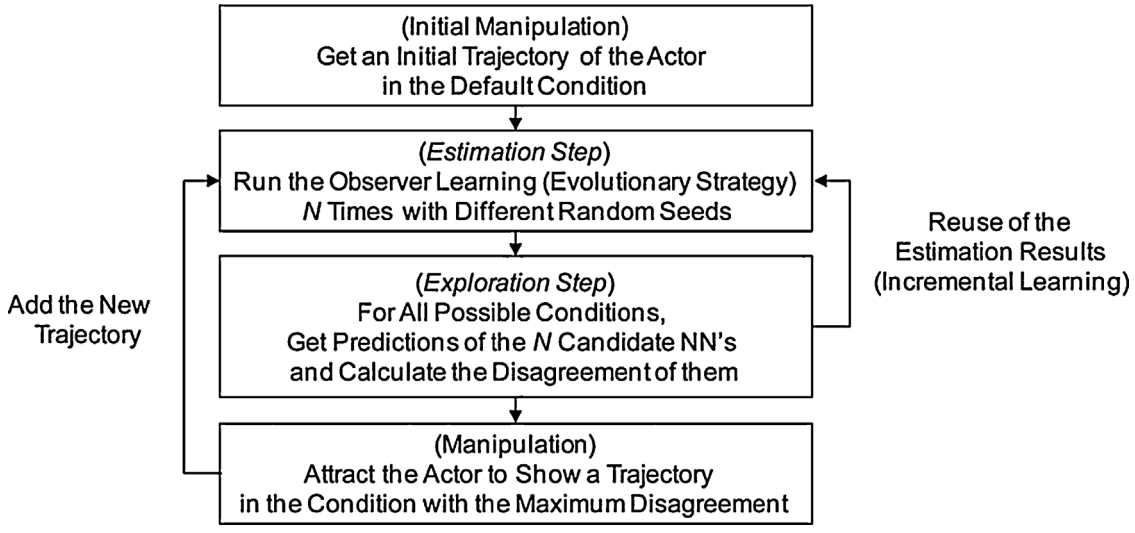


Fig. 3. The estimation–exploration algorithm (EEA) with evolutionary search.

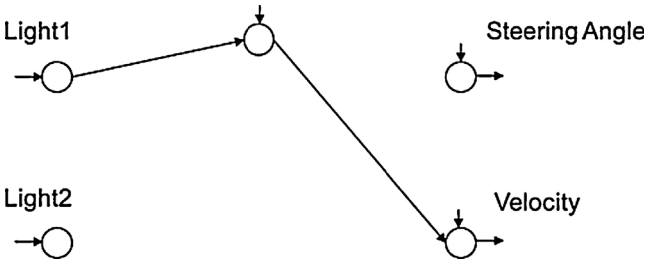


Fig. 4. An example of neural network created randomly.

with one of the input and output neurons. Each neuron has a unique ID number and each link stores two neuron ID numbers (for the source and destination neurons) with a weight value. Fig. 4 shows an example of the neural network created.

- Step 2: each neural network generates one offspring using one of the mutation operators, yielding $2 \times P$ neural networks (parents plus offspring). The fitness of each NN was evaluated based on the fitness function. We allowed only the fittest P NNs to survive to the next generation. In this paper, we used six types of mutation operators, with one of them being selected randomly.

1) Connection weight change: for a randomly selected link, the weight is reassigned as a new random value.
2) Bias change: it replaces the old bias with a new random value.
3) Link deletion: it deletes a link randomly if it is not the last link in the neural network.
4) Link addition: at first, it selects a hidden neuron randomly and inserts a new link for the node.
5) Hidden neuron addition: it creates a new hidden neuron with two connection links. One is from the hidden node and the other comes from the other node.
6) Hidden neuron deletion: it deletes one hidden neuron randomly if it is not the last hidden neuron.

- Step 3: repeat Step 2 until the maximum number of generations is reached.

3.3. Action exploitation

Once the actor's self-model was determined, several strategies could have been employed to exploit that knowledge. For example, a robot could change the position of the light source to elicit a desired behavior. The goal of this experiment was to determine a light position that would force the actor robot into a "trap" location. In the absence of any knowledge about the actor's controller, it is possible to make a straightforward guess that the actor will move in a straight line toward the light source. This strategy is known as "straight-line estimation (SLE)", and it served as our baseline for assessing the value of the NNs (see Fig. 5). Using the EEA, it is possible to make an estimate that may be superior to the straight-line estimation. In this research, the two approaches were compared for different trap positions by systematically changing the angle between the trap and the robot. For each trial, the distance between the virtual robot and the possible light source position was kept constant.

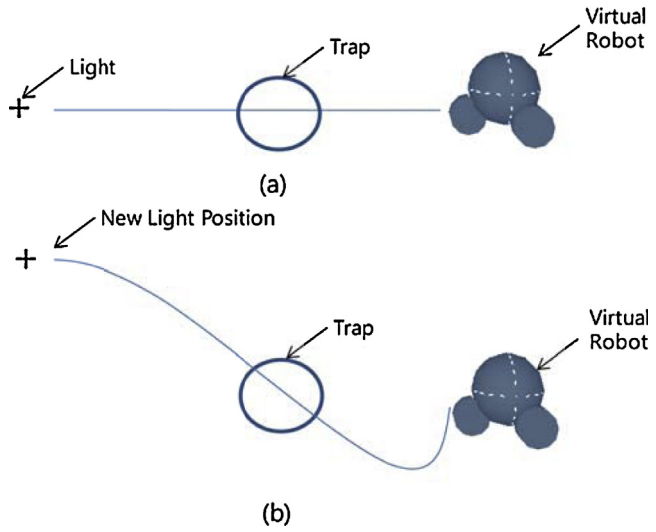


Fig. 5. Two different strategies to estimate the behavior of the actor robot. (a) Straight-line estimation assumes that the robot goes to light source in a straight line. (b) In ToM approach, the best light position is sought in order to force the actor into the "trap" location, based on the estimated NNs' prediction about the actor's trajectory.

4. Experimental results and discussion

In this research, we performed experiments using a virtual robot in physics-based simulation (PhysX) environments. The results were averaged over 10 runs.

4.1. Actor learning

In actor learning, the actor's neural controller evolves from a population of random neural networks (Fig. 6). In the actor learning for this research, the neural topology was comprised of 2 inputs, 3 hidden neurons, and 2 outputs with fully connected links.

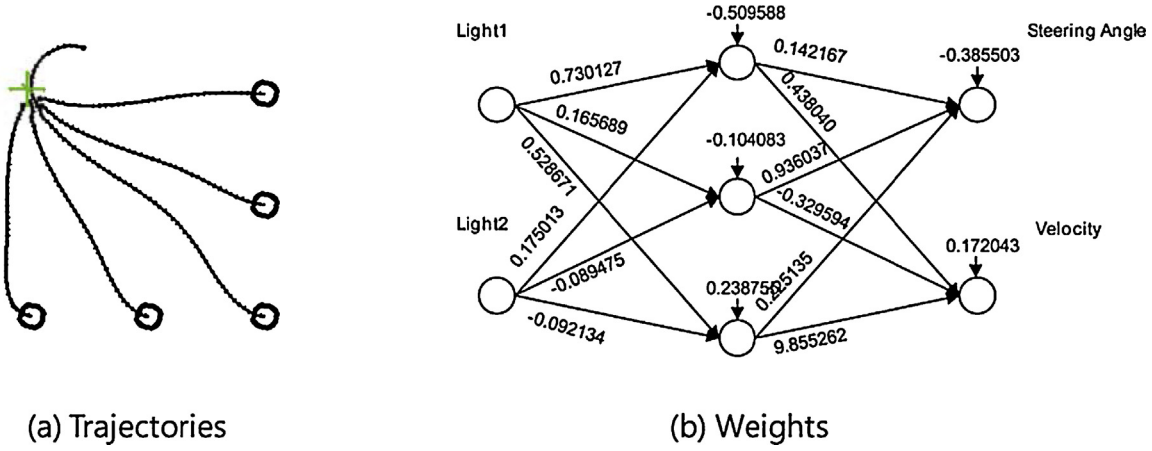


Fig. 6. Trajectories and weights of the actor's neural controller evolved (in (a), each circle represents the starting point of the actor robot and the green cross indicates a light source).

There were seventeen parameters that were trained in the evolution. The fitness value is the inverse of the sum of the distance between the current robot's position and the light source during the movement. In the evaluation, each robot was placed at (0, 0) and allowed to be simulated for 100 steps ($\Delta t = 1/6$ s between each step). Although the robot was trained from only the single point (0, 0), the final outcome showed that it followed the light sources in different positions. Because the purpose of this learning was not to determine an "optimal" neural controller for the light following task, the number of maximum generation was kept relatively small, at 50. The population size was set at 20.

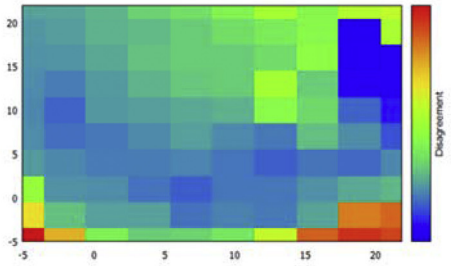
4.2. Observer learning with prior knowledge about others' internal models

In this case, the observer assumes that the actor has three hidden neurons with full connections, based on his brain structure. However, the observer has no information about the connection weights or the biases of the actor's NN. The observer needs to determine the appropriate weights of the network based on the actor's trajectories. In this research, the actor's initial angle was fixed at 0°. Therefore, the trajectory was dependent only on the starting position of the actor robot. For each cycle, the observer determined the desirable starting point and attracted the actor to that location. Subsequently, the actor followed the light source, showing their trajectory from that point. Because each trajectory is the result of the actor's demonstration, requests to the actor should be minimized.

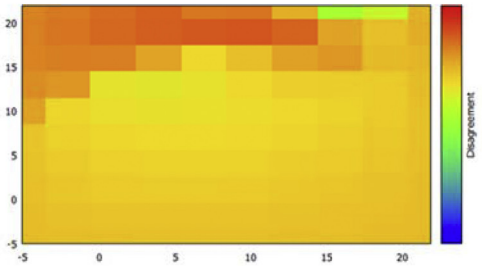
Initially, the observer attracted the actor to the center of the environment ($L\_X/2$, $L\_Y/2$). Using the first trajectory, the observer executed the "estimation" multiple times with different random seeds. In this paper, N, the number of independent "estimations" was five. For the "estimation", the observer initiated the population of neural networks with random weights [−0.2, 0.2]. In the learning stage, the observer co-evolved the self-adaptive parameters and the weights of neural network. The population size and the maximum number of generations were 20 and 200, respectively. The fitness function of the evolutionary search was defined as the similarity between the trajectories from the candidate neural network and the "observed" one.

After the estimation, the observer had N populations of neural networks. From each population, the observer selected the best one, based on the fitness value. At this point, the observer had N candidate neural networks. Using them, the observer selected the next probing point (starting point of the actor), based on the prediction disagreements of the candidates. In this research, the observer searched for the next starting point, from ([-5, 25],

(a) Possible starting points (resolution = 10x10) (each circle represents a starting position)



(b) Disagreement of predictions after the first cycle



(c) Disagreement of predictions from random neural networks

Fig. 7. Starting positions in the environment and their disagreement in terms of the predictions after the first cycle (for (b) and (c), the axis represents the x and y coordinates of the robot's starting position and the color level shows the disagreement). Blue indicates that the neural networks found show similar predictions about the trajectories from those locations. Conversely, red indicates that the prediction of trajectories by neural networks is significantly different. The results from the random neural networks had high disagreement; however, after observation, there is agreement related to predicting the actor's trajectories). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

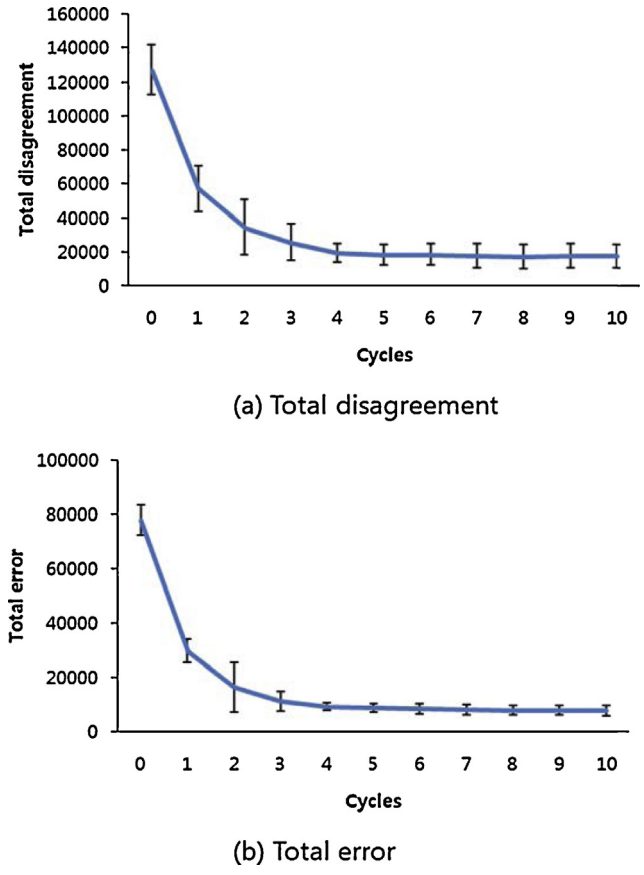


Fig. 8. The progress of the observer learning (cycle 0 indicates the initial random population of the neural networks).

[-5, 25]).Fig. 7(a) shows the possible starting positions. Because we used an exhaustive search to find the point with the maximum disagreement, we set the resolution as 1010 (for a total of 100 positions).

For each point, the observer predicted the trajectory of the actor starting there, using the N candidate neural networks. A disagreement between predictions about the point is defined as the sum of dissimilarity among the N trajectories. Fig. 7(b) shows this disagreement of predictions after the first cycle. This differs significantly from the disagreement map from random neural networks (Fig. 7(c)). This implies that the disagreement was high close to the (−5.0, −5.0) and (22.0, −5.0) starting points. In the center of the environment, there was relatively little disagreement, because the neural networks for the predictions were trained using the actor's trajectory from the center. In the next cycle, the observer attracted the actor to the (−5.0, −5.0), which had the maximum disagreement, to determine the second trajectory.

The observer thus had two trajectories, one from the center of the environment and the other from the point with the maximum disagreement after the first cycle. Using them, the observer executed the "estimation" N times. In the "estimation", the fitness function was defined as the similarity between the predictions and the observation for the two starting points. This process was repeated until the maximum number of cycles was achieved (for this research, there was a maximum of 10 potential cycles). The experiments were repeated 10 times, in order to determine accurate statistics. Fig. 8 shows the change in the total error and the disagreement over the cycles. The total error is defined as the sum of the difference between the actor's real trajectories and the predictions from the N candidates for the 100 starting positions. This was used to measure the actual progress of the observer learning (however, the information was not used in the learning).

The total disagreement is defined as the sum of disagreement among the N candidates' predictions for the 100 starting points. It is interesting to note that the total disagreement and the total error show similar curves over the cycles. The total error converges to

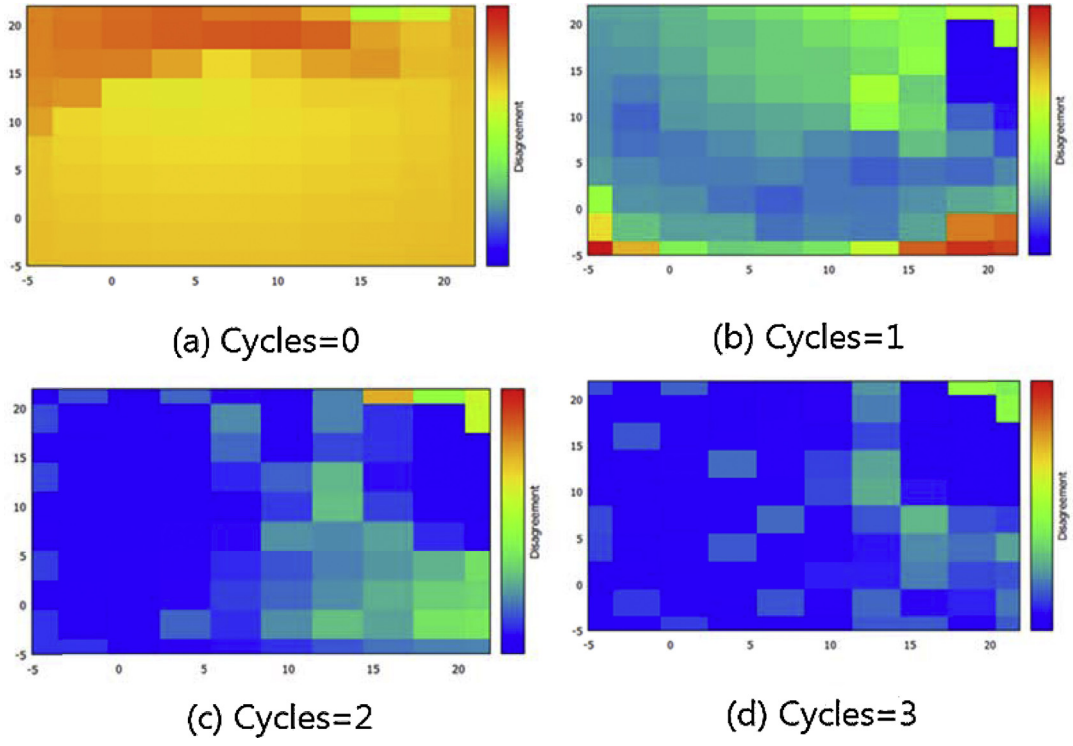


Fig. 9. The change in the disagreement over cycles.

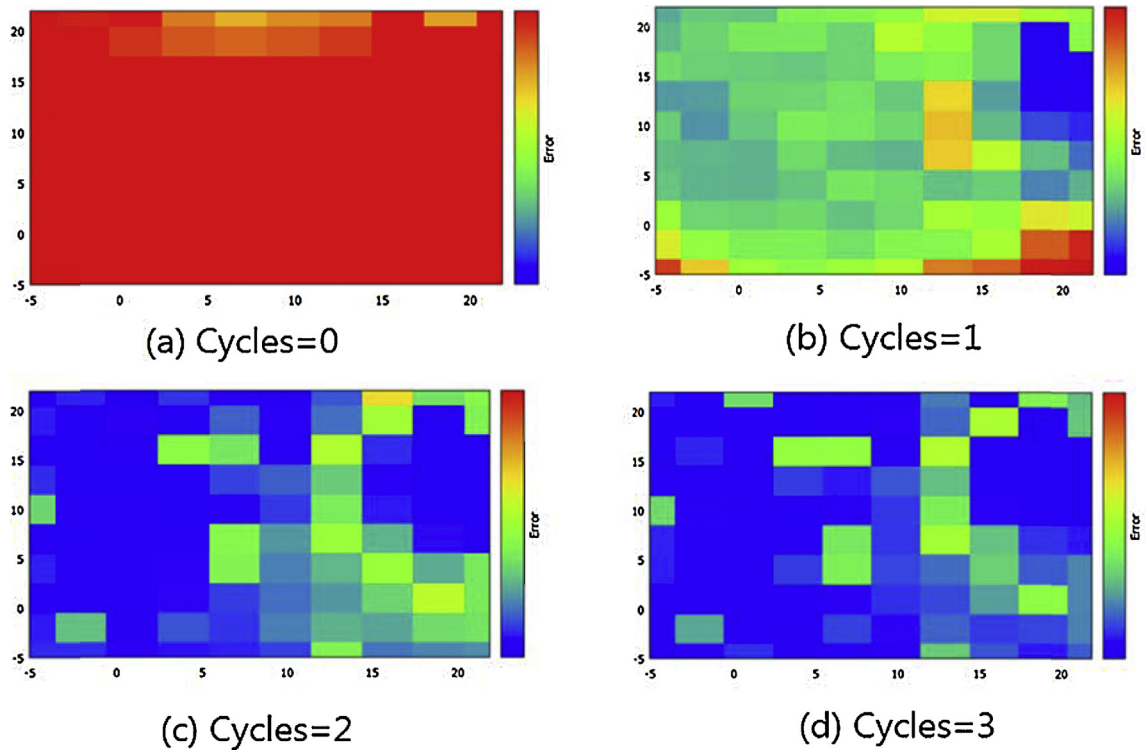


Fig. 10. The change in the errors over cycles.

the minimum only for five or six trajectories. Figs. 9 and 10 show the change in disagreement and error of each of the starting points for the cycles. There were similar patterns of change whose values decreased with subsequent cycles. Table 2 shows the total errors for the different experimental conditions, which changed the use of the incremental evolution (as the population of the previous cycles were copied onto the next cycle instead of using a random initialization) and active sampling (the next probing point was selected using the maximum disagreement of the predictions by N candidates). These results show that the two techniques are beneficial in observer learning in order to reduce error ($p = 0.05$).

In the exploitation stage, the observer exploits the neural networks that were determined using the observer learning. In our experiment, the goal of the exploitation was to catch the actor using a trap. In SLE (Straight Line Estimation), the observer predicts that the robot will go straight to the light source. The trap is placed in the middle (T_X, T_Y), between the robot's starting position (S_X, S_Y) and the light source (L_X, L_Y). Conversely, the observer with the ToM could predict the trajectory of the actor prior to the testing. It is possible to determine a new position of the light source (L_X2, L_Y2) in order to force the actor into a trap. The exploitation error of this approach can be measured as the minimum distance between the trap and the actor robot.

Fig. 11 shows a successful example of this exploitation. In this example, the robot was placed at (0, 0) and the light source was in different positions. In the SLE approach, the robot went straight through the trap; however, there could be error from the simple guess when the actor follows the light source. In the ToM approach, the observer robot searched for a new light source position in order to force the actor to fall into a trap. In the SLE, the distance was 20, between the starting position and the light source. Therefore, the new light source position was limited to a point on a circle (radius = 20) centered at the starting point. The observer simulated the actor's trajectories with 100 potential new light source positions on the circle using the neural network obtained. From this simulation, the observer determines the best new light source position. As seen in Fig. 11(c), the new light position was estimated as (18.6, −7.4) and, as a result, the actor went very close to the trap. In this example, the errors were 3.85 for SLE and 0.14 for ToM. We compared the two approaches for different angles (0–2π). For 100 different angles, the SLE error was 2.07 ± 1.26 and the ToM error was 0.39 ± 0.43 ($p < 0.001$).

Table 2
Total error with different experimental conditions and statistical test results (t-test) (IE: incremental evolution, AS: active sampling).

Symbol	Prior knowledge	Incremental evolution	Active sampling	Total error (after 10 cycles)
Prior	Yes	–	–	11336 ± 4015
Prior + AS	Yes	–	Yes	10613 ± 2292
Prior + IE	Yes	Yes	–	9419 ± 2902
Prior + IE + AS	Yes	Yes	Yes	7912 ± 1913
IE + AS	–	Yes	Yes	12772 ± 3482

	Prior + IE + AS
Prior	$p < 0.05$
Prior + AS	$p < 0.02$
IE + AS	$p < 0.002$

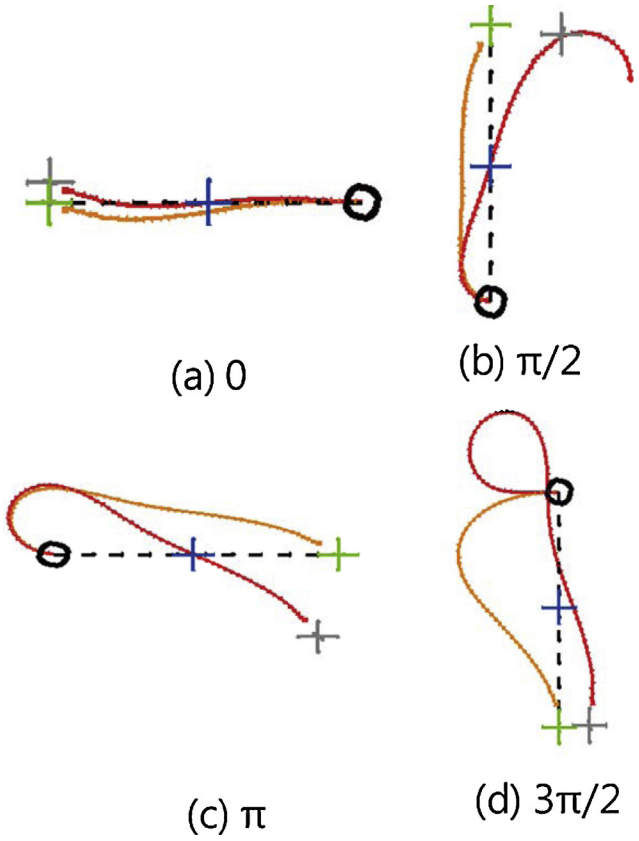


Fig. 11. An example of successful exploitation using the ToM approach (the dotted line shows the estimation by SLE. The cross in the middle of the dotted line is the trap. The circle represents the starting points of the robot. The observer determined the new light position (the gray cross) in order to force the actor to go to the trap. The solid lines show the trajectory of the actor).

4.3. Observer learning without prior knowledge on others' internal models

Because the observer has no information on the internal topology of the actor's neural network, the target of the optimization includes both the topology and their connection weights. This is done using a very simple network topology, then increasing its complexity by adding hidden nodes and connections gradually. To limit the size of the neural network, the maximum number of nodes (M) is defined. In this paper, we compared the observer learning for $M = 10$, 20, 30, 40, 50, and ∞ (Fig. 12). Although the original actor's NN had seven nodes (2 input nodes + 3 hidden nodes + 2 output nodes), the observer learning showed the lowest total error ($12{,}772 \pm 3482$) at $M = 30$. Compared

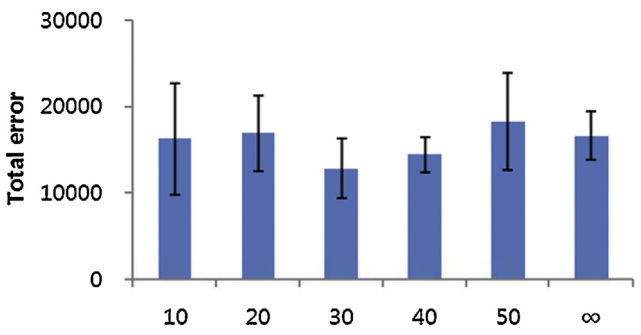


Fig. 12. The comparison of the total error on different M (the maximum number of nodes).

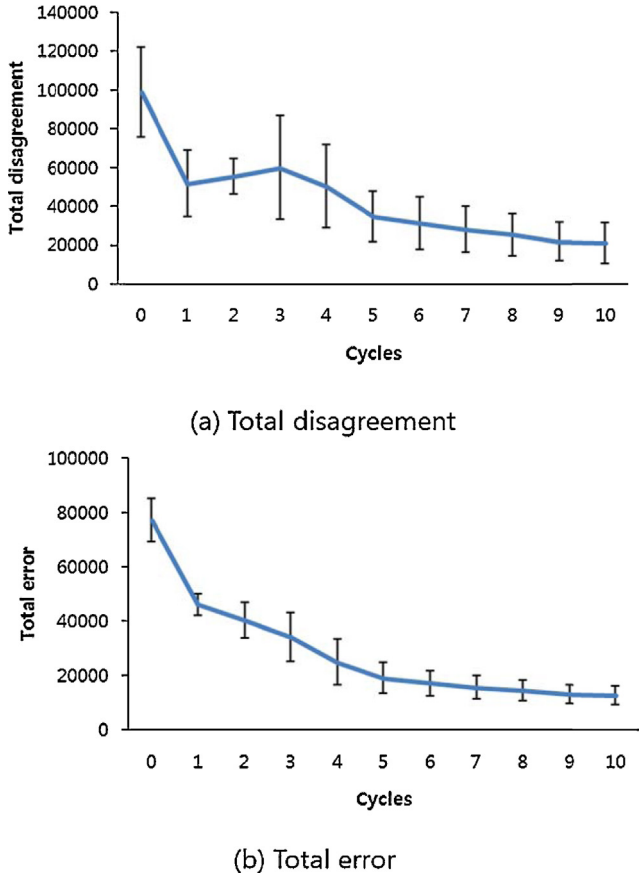


Fig. 13. The progress of the observer learning (the maximum number of nodes = 30).

to the observer learning with topology information (7912 ± 1913), the total error was relatively high ($p < 0.002$). Fig. 13 shows the changes in the disagreement and the total error for each of the cycles. In the exploitation, for 25 different angles (0–$\pi/2$), the SLE error was 1.28 ± 0.41 and the ToM error was 0.72 ± 1.34. However, there was no improvement for the angles ($\pi/2$–2π).

5. Concluding remarks

We used a reverse-engineering algorithm to construct a model of the internal neural network of robots using observations of their behavior. The observer robots actively collected information on the actor's trajectory toward a goal and inferred an internal model based on the behavior. A series of experiments showed that the proposed method can be useful in identifying internal models. Furthermore, this research demonstrated the possibility of using a neural-based approach as a platform for learning cognitive functions. Initially, the observer had a limited knowledge of the actor, and it actively induced the actor to show behaviors, from which the observer constructed a self-model of the actor. The observer was then able to exploit the internal model in order to predict the future behaviors of the actor. In particular, we tested the use of prior knowledge for the inferences, determining that the observer was capable of inferring the other's internal model with a low rate of error, even without this information.

Using the robotic ToM framework, we conducted psychological experiments similar to human tests by changing the experimental conditions. This approach yields results comparable to those from human tests, and it has the advantage of allowing experiments to be carried out that might not be feasible with human subjects. In the context of robot research, the implementation of ToM is

important in the development of social robots. This study shows that the robotic ToM platform can be used to investigate various issues in several disciplines. For example, the visibility of the observation can be used to study the robustness of ToM and machine learning. It is also possible to test several conditions that may affect ToM when information is incomplete or distracted (Kim et al., 2013).

It is, in principle, possible to extend the results from physics-based simulations to a robot platform, such as the one described here. For example, motion-tracking devices can be used to track the movement of mobile robots, and EEA can be used to infer the internal models of real robots. In this study, we used the feed-forward neural network as an internal neural model of an actor. This work should be extended to include recurrent neural network models strong for temporal processing.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIP) (2013 R1A2A2A01016589, 2010-0018948, 2010-0018950). The authors would like to express thanks to Prof. Hod Lipson for his guidance on the early version of this work.

References

Baron-Cohen, S., 1995. Mind Blindness: An Essay on Autism and Theory of Mind. Cambridge, MIT Press.
Bongard, J., Lipson, H., 2007. Automated reverse engineering of nonlinear dynamical systems. Proc. NatL. Acad. Sci. 104 (24), 9943–9948.
Bongard, J., Zykov, V., Lipson, H., 2006. Resilient machines through continuous self-modeling. Science 314 (5802), 1118–1121.
Bosse, T., Memon, Z.A., Treur, J., 2007. A two-level BDI-agent model for theory of mind and its use in social manipulation. Proceedings of the Artificial and Ambient Intelligence Conference 335–342.
Breazeal, C., Buchsbaum, D., Gray, J., Gatenby, D., Blumberg, B., 2005. Learning from and about others: Towards using imitation to bootstrap the social understanding of others by robots. Artif. Life 11, 31–62.
Bringsjord, A., Shilliday, D., Werner, M., Charpentier, E., Bringsjord, A., 2008. Toward logic-based cognitively robust synthetic characters in digital environments. Proceedings of the First Artificial General Intelligence 87–98.
Buchsbaum, D., Blumberg, B., Breazeal, C., Meltzoff, A.N., 2005. A simulation-theory inspired social learning system for interactive characters. IEEE International Workshop on Robots and Human Interactive Communication 85–90.
Call, J., Tomasello, M., 2008. Does the chimpanzee have a theory of mind? 30 years later. Trends Cogn. Sci. 12 (5), 187–192.
Demiris, Y., Johnson, M., 2003. Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. Connect. Sci. 15 (4), 231–243.
Floreano, D., Keller, L., 2010. Evolution of adaptive behavior in robots by means of Darwinian selection. PLOS Biol.(Jan) .
Freund, Y., Seung, H.S., Shamir, E., Tishby, N., 1997. Selective sampling using the query by committee algorithm. Mach. Learn. 28, 133–168.
Herrmann, E., Call, J., Hernandez-Lloreda, M.V., Hare, B., Tomasello, M., 2007. Humans have evolved specialized skills of social cognition: the cultural intelligence hypothesis. Science 317, 1360–1366.
Kaliouby, R.E., Robinson, P., 2004. Mind reading machines: automated inference of cognitive mental states from video. IEEE International Conference on Systems, Man and Cybernetics 682–688.
Kelley, R., King, C., Tavakkoli, A., Nicolescu, M., Nicolescu, M., Bebis, G., 2008. An architecture for understanding intent using a novel hidden markov formulation. Int. J. Hum. Robot. 5 (2), 1–22.
Kim, K.-J., Cho, S.-B., 2012. Automated synthesis of multiple analog circuits using evolutionary computation for redundancy-based fault-tolerance. Appl. Soft Comput. 12 (4), 1309–1321.
Kim, K.-J., Lipson, H., 2009a. Theory of mind in simulated robots. Genetic and Evolutionary Computation Conference (GECCO) – Late Braking Papers 2071–2076.
Kim, K.-J., Lipson, H., 2009b. Towards a simple robotic theory of mind. Performance Metrics for Intelligent Systems Workshop 131–138.
Kim, T.-S., Na, J.-C., Kim, K.-J., 2012. Optimization of autonomous car controller using self-adaptive evolutionary strategy. Int. J. Adv. Robot..
Kim, K.-J., Eo, K.-Y., Jung, Y.-R., Kim, S.-O., Cho, S.-B., 2013. Evolutionary conditions for the emergence of robotic theory of mind with multiple goals. IEEE Workshop on Robotic Intelligence in Informationally Structured Space (RiiSS) 48–54.
Kondo, K., Nishikawa, I., 2003. The role that the internal model of the others plays in cooperative behavior. Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication 265–270.
Nolfi, S., Floreano, D., 2004. Evolutionary Robotics. MIT Press.
Ozonoff, S., Pennington, B.F., Rogers, S.J., 1991. Executive function deficits in high functioning autistic individuals: Relationship to theory of mind. J. Child Psychol. 32, 1081–1151.
Peters, C., 2006. A perceptually-based theory of mind for agent interaction initiation. Int. J. Hum. Robot. 3 (3), 321–339.
Premack, D.G., Woodruff, G., 1978. Does the chimpanzee have a theory of mind? Behav. Brain Sci. 1, 515–526.
Pynadath, D.V., Marsella, S.C., 2005. PsychSim: Theory of mind with decision-theoretic agents. Proceedings of the International Joint Conference on Artificial Intelligence 1181–1186.
Saxe, R., 2009. Theory of mind (neural basis). Encyclopedia Consciousness 401–409.
Scassellati, B., 2002. Theory of mind for a humanoid robot. Auton. Robots 12 (1), 13–24.
Siegal, M., Varley, R., 2002. Neural systems involved in 'theory of mind'. Nat. Rev. Neuroscience 3, 463–471.
Takanashi, T., Asada, M., Negrello, M., 2007. Emulation and behavior understanding through shared values. IEEE/RSJ International Conference on Intelligent Robots and Systems 3950–3955.
Takano, M., Arita, T., 2006. Asymmetry between even and odd levels of recursion in a theory of mind. Proceedings of ALIFE X 405–411.
Webb, B., 2001. Can robots make good models of biological behavior. Behav. Brain Sci. 24 (6), 1033–1050.
Wischmann, S., Floreano, D., Keller, L., 2012. Historical contingency affects signaling strategies and competitive abilities in evolving populations of simulated robots. Proc. National Acad. Sci. 109 (3), 864–868.
Yokoya, T., Ogata, J., Komatani, K., Okuno, H.G., 2007. Discovery of other individuals by projecting a self-model through imitation. IEEE/RSJ International Conference on Intelligent Robots and Systems 1009–1014.
Youmans, G.L., 2004. Theory of mind in individuals with Alzheimer – type dementia profiles. Ph.D. Thesis. College of Communication at the Florida State University.
Zanlungo, F., 2007. A collision-avoiding mechanism based on a theory of mind. Adv. Complex Syst. 10 (2), 363–371.