J Intell Robot Syst (2012) 66:167–182
DOI 10.1007/s10846-011-9624-y

Modelling Shared Attention Through Relational Reinforcement Learning

Renato Ramos da Silva · Roseli Aparecida Francelin Romero

Received: 3 December 2010 / Accepted: 25 July 2011 / Published online: 18 August 2011

Abstract Shared attention is a type of communication very important among human beings. It is sometimes reserved for the more complex form of communication being constituted by a sequence of four steps: mutual gaze, gaze following, imperative pointing and declarative pointing. Some approaches have been proposed in Human−Robot Interaction area to solve part of shared attention process, that is, the most of works proposed try to solve the first two steps. Models based on temporal difference, neural networks, probabilistic and reinforcement learning are methods used in several works. In this article, we are presenting a robotic architecture that provides a robot or agent, the capacity of learning mutual gaze, gaze following and declarative pointing using a robotic head interacting with a caregiver. Three learning methods have been incorporated to this architecture and a comparison of their performance has been done to find the most adequate to be used in real experiment. The learning capabilities of this architecture have been analyzed by observing the robot interacting with the human in a controlled environment. The experimental results show that the robotic head is able to produce appropriate behavior and to learn from sociable interaction.

Keywords Shared attention · Relational reinforcement learning · Social robotics

R. R. da Silva (✉) · R. A. F. Romero
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo,
Campus de São Carlos Caixa Postal 668,
13560-970 São Carlos, SP, Brazil
e-mail: ramos@icmc.usp.br

R. A. F. Romero
e-mail: rafrance@icmc.usp.br

1 Introduction

Recently, shared attention has received a increasing interest, due to its importance in the development of interactions and communications. Some works have focused on modeling and understanding human developmental processes [35], others only use it as a part of interaction process with sociable robot. Overall, we can find several architectures in literature with this purpose [3, 29, 38].

The observable behaviors of an individual attending to an object or event that other person observes at a given instant can be regarded as shared attention [38]. A formal definition can be made by a sequence constitutes by four steps [12]:

1. mutual gaze—this occurs when the establishment of eye contact between two people

happens, that is there is an attention between them
2. gaze following—this occurs when one person intentionally look at an object of interest or some event, by the which another person is looking at.
3. imperative pointing—it is a gaze following followed by a pointing to an object or event.
4. declarative pointing—this consist in the three first steps with some assumption about the scene in an appropriate sociable context

In addition, there is the term "joint attention" that is found in the literature and causes some confusion with shared attention. A process can be considered joint attention when it is comprised of a sequence of mutual gaze and gaze following. Many researchers have focused on joint attention or gaze following, because these processes are simpler then the complete shared attention process. However, Scasselatti used an upper-torso humanoid robot with a mechanism for non-linguistic mechanisms of shared attention [30], in which, he proposed an approach for human-robot interaction involving the three first steps.

This paper outlines a proposal for improving a robotic architecture proposed in our previous work [29] in order it could be inserted in a robotic head. It was evaluated in a controlled environment for learning the functionalities mentioned in the three steps of shared attention: mutual gaze, gaze following and declarative pointing. Thus, it can be considered as a step forward to reach the process described above.

This article is organized as it follows. We start introducing some related works in Section 2. Then, the robotic architecture is presented in Section 3. Afterward, in Section 4, three learning methods, incorporated into learning mechanism module of the robotic architecture, are presented. In the Section 5, it is presented a comparison among the three learning methods in a sociable interaction simulator to find the most adequate to be used in real experiment. Further, using the learning method chosen, a set of experiments is carried out to evaluate the performance of the robotic architecture controlling a robotic head. Finally, in the Section 6, are presented the conclusions and future works.

2 Related Works

One of first papers in the literature that utilized the shared attention mechanism was proposed by Scasselati [30, 31]. Recently, several Human-Robot Interaction—HRI studies evaluated part of shared attention process on robots. We can divide in two groups. The basic difference between them is that one is focused on HRI where shared attention is used for turning the interaction more natural and the other turns the robot able to learn shared attention. In the last case, it is worth to note that only a part of shared attention process.

Some works related to the first group are described to follow. A model of human gaze was proposed to be used on a humanoid robot for creating a natural, human-like behavior for storytelling [21]. In [41], it was performed experiments with a guide robot designed using data from human experiments to turn its head towards the audience at important points during its presentation. A study of the importance of eye-tracking to give joint attention to the robot from human eye gaze is found in [36]. Mutlu et al. [22] studied how a robot can establish the participant roles (address, bystander, and overhearer) of its conversational partners using gaze cues [42].

Some works belonging to the second group include the use of method such as temporal-difference (TD) reinforcement learning scheme for learning joint visual attention [16]. This model is limited in the sense that the robot only gets reward when the object, operated by the observer, moves itself. Also the caregiver's face is treated separately from the objects and does not lead to any reward, that is, mutual gaze was not considered in this work. Nagai [24] and Nagai et al. [23] used face edge features and motion information (optical flow) to estimate the sensor motor coordination and the motor output using two separate neural networks. Their model does not utilize the depth information of the images and thus can not handle ambiguous situations where an object appears in robot's gaze direction that may not be located within the caregiver's gaze direction. Shon et al. [32] presented a probabilistic model of gaze following imitation in which estimated gaze vectors are used in conjunction with the saliency maps of the visual scenes to produce maximum

a posteriori (MAP) estimates of the object positions attended by the caregiver. Triesh proposed a basic set of structures and mechanism for gaze following. This set includes perceptual skills and preferences, reward-driven learning, habituation and a structured social environment. This work is evaluated only on simulator. Kim et al. [14] improved a model that uses a basic set [38], on a robotic head. They have been used an actor-critic reinforcement learning model for learning gaze following. The drawback of the proposed method is that a salient map is used as additional information well as representations of the caregivert's head direction (h) and the caregivert's eye direction (e). In our previous work [29], we applied the contingency learning in an architecture for joint attention aiming to control a real robotic head.

All works mentioned above use mutual gaze and gaze following, but we will present an architecture that learns declarative pointing as a part in the process of shared attention and this architecture will be used for controlling a robotic head.

3 Robotic Architecture

The robotic architecture to be extended in the present work was proposed by Policastro [28] and Policastro et al. [29]. It is constituted by mechanisms and structures inspired on Science of Behavior Analysis [5, 6, 27]. It is composed by the following modules:

- Stimulus Perception that acquires and codifies the state of the environment
- Consequence Control that simulates internal necessities for a pro-active interaction
- Response Emission that contains a learning mechanism for choice an action with input data from other modules

Figure 1 illustrates the general organization of the architecture and the interaction among these three main modules. In this figure, arrows indicate the flow of information in the three modules of the architecture. The circles indicate the methods and component structures of the modules [29].

During an interaction, the Stimulus Perception module acquires and codifies the state of the environment and deploys this coded state for the Response Emission and the Consequence Control modules. After, the Consequence Control module checks the internal state of the robot and sets the active necessities, if there is one. Then, all information is used by Response Emission to select an action. Afterward, the selected response is emitted by executing a motor script. Finally, the Stimulus Perception module acquires and encodes the new current environment state and the architecture control enters a loop that may be finished either at the end of an interaction or when the robot reaches its goal [29].

In addition, the architecture also employs a working memory to exchange all information among the three main modules. This memory is used to keep information about stimuli (antecedents and consequents), the last emitted response and internal necessities. Each element inserted into the working memory has a counter that keeps the notion of time. When a new element is inserted in the working memory, its age counter is set to zero and it is incremented by 1 whenever new subsequent predicates are inserted. So, elements persist by a number of time steps in the memory. This mechanism is employed to control the chronology of facts and events [29].

A voice recognition system is based on the Nuance solution [25] and it is able to recognize naturally spoken *Portuguese* utterances. It contains a speech recognizer and grammatical knowledge base. The speech synthesis is performed by joining pre-recorded prompts in order to build complete phrases. This strategy enables short conversations with the robot [29]. This module is responsible by the ability of our architecture be able to do a declarative pointing.

We describe the main structures and modules of our architecture as it follows.

3.1 Stimulus Perception Module and Knowledge Representation

The stimulus perception module employs algorithms of vision system based on the work proposed by Breazeal and Scassellati [4]. It is composed by two mechanisms: the face recognition with head pose estimation (based on Adaptive

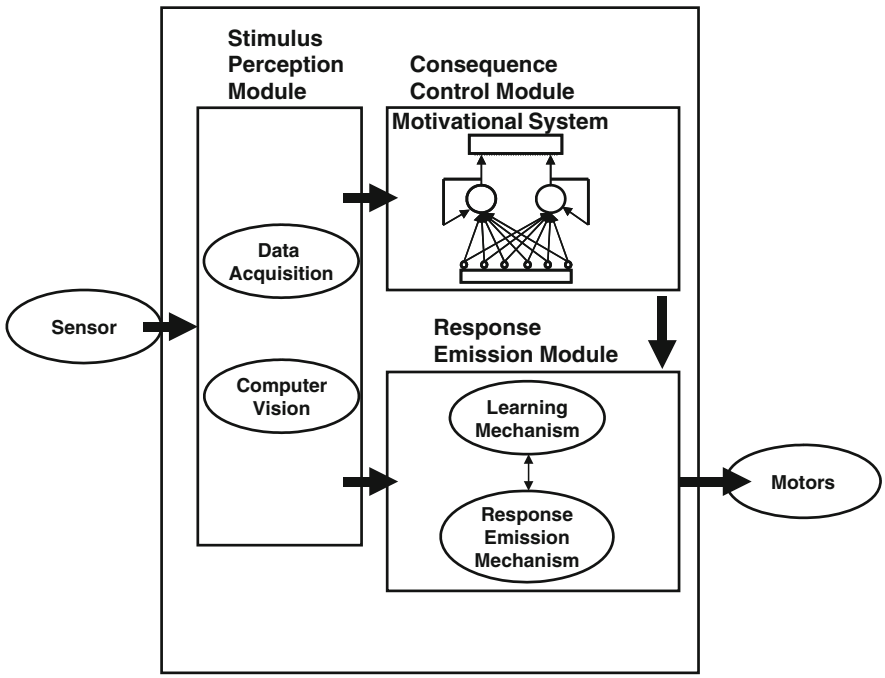

Fig. 1 General organization of the architecture [29]

Appearance Model [19]) and the visual attention mechanism (based on saliency [11]).

The implementation of the vision system is based on maps of characteristics processed for each perception (colors and faces). The vision system creates an activation map that can be used by the other modules of our architecture in order to control the robot's behavior [29].

The color map is based on visual attention presented by Itti and their colleagues in [11]. This process employs a biologically inspired visual attention mechanism to create a map of characteristics that represents the *visual saliency* of the scene. This visual saliency is formed by the composition of several maps of characteristics extracted from the image. Each map of characteristics presents an elementary property of the image as color, intensity and orientation. These characteristics are known as primitive visual characteristics. The method for the construction of this saliency map can be divided into five stages: extraction of characteristics, linear filtering, center-surround difference calculations, sum of the maps of characteristics (liner combination) and selection of salient areas. The selection of the most salient area is carried out using a saliency threshold and a minimum length of area threshold. Afterwards, a process based on color histograms is carried out to obtain the more frequent values of the channels r, g, b (of the RGB color space) and h (of the HSI color space) in the area of interest. This color map was developed using the functions of saliency of the Lti-Lib library [15, 29].

The face map is based on works presented by Morency [19], about face and pose detection. The face detection is carried out employing an approach based on active appearance model [19]. In this approach, the principal component analysis (PCA) is used for find the vectors that best describe the distribution of images inside the whole space of training images. Once a face is detected, the vision system proceeds the detection of its pose (*pan* and *tilt* angle). This algorithm creates a reference model using an initial frame and calculates the changes of the pose employing the created model. This face map was developed using the functions of face detection of the Watson library [19, 29].

Thus, this module detects the state from the environment and encodes this state using an appropriate representation. The knowledge representation adopted for the architecture is based on

a relational representation [7, 26], enabling the representation of large spaces in an economical way. This representation is also know of first order logic and we adopt the same logic bases of [20].

The architecture encodes knowledge as stimuli, facts, responses, behavior rules and constraint rules, which are represented by associating each stimulus and fact with a binary number. This association is made by a function, $f = 2^x$, in which x ranges from 0 to $n - 1$, and n is the total number of stimuli and facts. This function allows us to represent the state of environment by a binary sum and we can separate it without lose any fact.

This simple technique is also known as binarization by conversion of a categorical attribute to asymmetric binary attributes [37].

Stimuli encode all signals received from the environment and they are represented as atoms or objects that may have properties like color, size, shape, position and pose (for faces). These atoms represent objects detected from the real world, relevant in the problem domain. Properties of the stimuli are set by the Stimulus Perception Module, in order to encode the current environment state. For example, one may define *face* as a stimulus of the environment to be detected and set a routine to encode *skin color* in its color property, its position and pose angles (*pan* and *tilt*). The environment state is encoded employing perception predicates: *see(X)*, *hear(Y)*, *at(Z)* and *smell(W)*. The perception predicates relate all detected stimuli to build a representation from the current environment state [29].

3.2 Consequence Control Module

An artificial motivational system may enable a robot to pro-actively interact with the environment, driving its behaviors to satiate its artificial internal necessities. The consequence control module is composed by a motivational system that simulates internal necessities of an individual. The motivational system is based on works presented by Breazeal [3] and Gadanho [8] and it has one or more necessity units implemented as a simple perceptron with recurrent connections [10].

The system is modeled as a competitive artificial neural network with recurrent connections. *Stimulus* is a set of input stimulus from the environment. *Preprocessor* encodes the input signals received from the environment into an appropriate form. Units *i*, *j*, ...*m* represent an array of *m* necessity units that can be simulated by the system. *Bias* is the activation bias of each unit. The output y of each unit is given by $y = f(u)$. *Mediator* selects the dominant necessity, that is higher than a predefined *threshold*. Figure 2 shows the general architecture of the motivational system [29].

The activation of a necessity unit is given by:

$$u = \left(\sum_{j=1}^{n} w_j \times i_j \right) + w_r \times i_r + b \tag{1}$$

where i_j is the input signal representing a stimuli detected from the environment, i_r is the signal from the recurrent connection, w_j are connection weights of the input signal, w_r is connection weight of the recurrent signal, and b is the bias of the unit. All weights and bias are empirically defined according to the necessity being simulated. The output of a necessity unit is given by:

$$y = \frac{1}{1 + e^{-(u+\delta)}} \tag{2}$$

where u is the activation value and δ is the sigmoid function inclination. A necessity unit simulates the internal necessities of an individual. Additionally, the motivational system has an output mediator that mediates activation among several necessity units, employing competition and an activation threshold [29].

The motivational system works as it follows. Initially, the stimulus detected from the environment are sent to the consequence control module. Then, the *Preprocessor* encodes these stimulus to construct an appropriate input pattern. This input pattern may be or not normalized, depending on the numeric interval of the selected connection weights and problem domain. Afterwards, the necessity units calculate their activation values employing the Eq. 1, and their output values employing the Eq. 2. After this, the *Mediator* performs a competition among all unit outputs and selects the winner. The *Mediator* checks if the winner is higher than the activation threshold. If so, the motivational system outputs the active necessity.

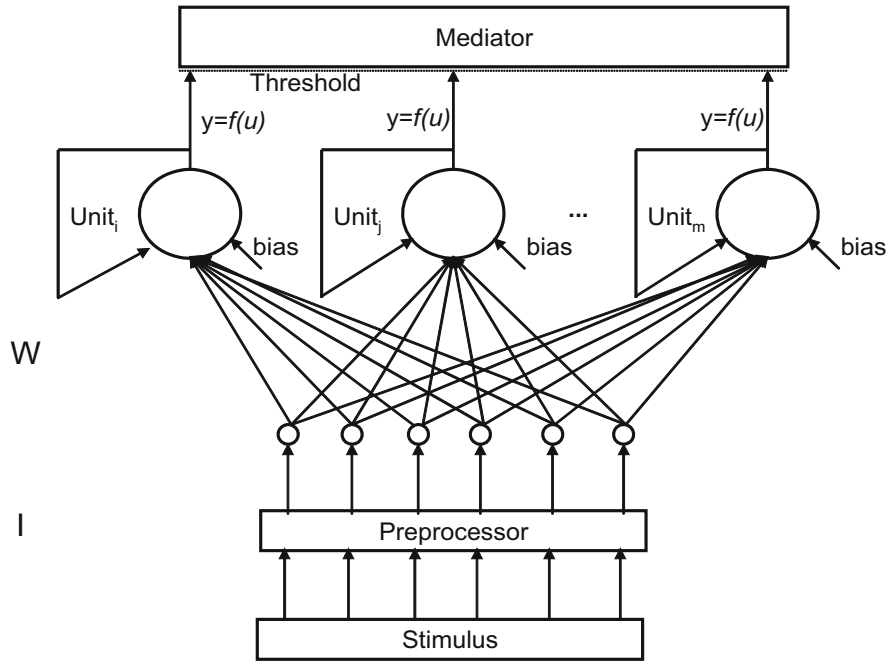

Fig. 2 General architecture of the motivational system [29]

After this, the motivational system checks and reports if any necessity unit has got a reinforcement [29].

3.3 Response Emission Module

The response emission module is composed by a learning mechanism and response emission mechanism. The first one constructs a nondeterministic policy for response emission, that is, what response is to be emitted on the presence of certain antecedent stimulus. These stimuli are information stored at work memory that come from other modules.

The response emission mechanism takes the information from learning mechanism and send it to the actuators to be executed.

3.4 A Robotic Head

In this section, we briefly present the robotic head showed on Fig. 3. This interactive robotic head is composed of 5 servo-motors and two color web cameras (one camera used for face tracking and one camera used for object detection). This mechanism enables the robot head to move in 6 different directions (left, left down, down, right down, right and center). At moment when the architecture has been executed into computer, the communication with robotic head is made by a serial cable.

The motor system of the interactive head has several motor scripts of engine that allows the robot to look forward (when it wants to interact with a human) and to look to several places in the environment (to find objects).

The basic operation of the architecture can be summarized as follows: during an interaction, the

Fig. 3 Robotic head—WHA8030 of Dr. Robot [1]

stimulus perception module acquires and codifies the environment state. It is send to others modules: the response emission and the consequence control. After, the consequence control module checks the internal state of the robot to make two operations: sets the active necessities, if there is one and generate a reward on the basis of the internal state estimate. Then, learning mechanism take the best action for that state that meets certain needs. Afterward, the selected response is emitted by executing a motor script. Finally, the architecture control enters a loop that may be finished at the end of an interaction.

In fact, the necessity associates the state with actions look upon shared attention problem. We have used two necessities: none and attention. Each one is associated with a pair (action,state) in training phase of the learning algorithm. When the architecture searches for an action, it verifies if the necessity value is equal to that one produced by the consequence control module. Only actions with the same necessity value are candidate to be chosen. Then, the reinforcement value is used to choose the best action. This mechanism is important to reinforcement learning operation because the problem of share attention can be addressed in a fully observable state.

4 The Learning Mechanism

In this section we present three learning methods that were incorporated into the robotic architecture. They are: Contingency, Q-learning and ETG methods.

4.1 Contingency

The proposed contingency learning is carried out through a nondeterministic reinforcement learning algorithm by storing new behavior rules and updating the execution probability of existing ones [29].

The response emission module uses the state and necessity information to select a response to be emitted by the robot. Response selection is done in a probabilistic way, based on Roulette Wheel selection method [9]. This method is also called stochastic sampling with replacement. The roulette-wheel selection algorithm provides a zero bias and the probability to be chosen are proportional to the fitness value. All responses in the robott's repertory keeps a default fitness value that is predefined as a parameter in the architecture. This default fitness value, as well as fitness values from the behavior rules, is employed for building the selection roulette. While selecting the appropriate behavior rules, the response selection method can increase or decrease its fitness value by an influence rate, if a rule satisfies an active necessity, or if a rule satisfies an inactive necessity, respectively [29].

Afterwards, the selected response is emitted by executing a motor script. If the last emitted response is not yet a rule, the learning algorithm then links the three-term of the contingency (antecedent stimulus, last emitted response and consequence), storing this new knowledge as a new behavior rule. If the behavior rule already exists, the architecture updates its fitness using the perceived consequence of its execution [29].

4.2 Q-learning

The Q-learning algorithm, a technique proposed by Watkins [39], is an iterative method for learning a policy of action in autonomous agents. It is one of the most used algorithms belonging to the reinforcement learning paradigm due to its simplicity of implementation. It is known to be an algorithm that computes the temporal difference Q value (Q) for an optimal policy associated with each pair (state, action), being built by an arbitrary exploitation of the environment [39].

In Q-learning algorithm, the agent learns a Q-function to estimate the value of taking an action from a state. An agentt's policy is typically to take the action with the highest Q-value in the current state, except for occasional exploratory actions. After taking the action and receiving some reward (possibly zero), the agent updates its Q-value estimates for the current state.

Q-learning [39], an off-policy learning algorithm or it learns by exploring the space arbitrarily, falls within the "model-free" branch of the reinforcement learning family, because it does not require the agent to learn a model of the world or environment (i.e. how actions lead from state

to state, and how states give rewards) in order to learn an optimal policy. Instead, the agent interacts with the environment by executing actions and perceiving the results they produce. In Q-learning algorithm, each possible state-action pair is assigned a quality value, or "Q-value" and actions are chosen by selecting the action in a particular state which has the highest Q-value according to Eq. 3:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_t \left[r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \right] \quad (3)$$

in which the learning rate (α) varies between 0 and 1 ($0 < \alpha < 1$) and γ is the discounter parameter ($0 < \alpha < 1$) [18].

The *Q-learning* algorithm that we have used in the architecture is showed on Algorithm 1. We use the basis of *Q-learning* algorithm proposed by Watkins [39], but we include the necessity factor (n_i) and it is storage on Q-table in the algorithm as a factor required by the architecture.

Algorithm 1 Q-Learning Algorithm

```
initialize Q(s, a) arbitrarily
i ← 0
repeat
   take state s_i
   take necessity n_i
   choose a_i for s_i using a policy derived from
   the current hypothesis Q
   take action a_i, observe r_i and s_{i+1}
   Update Q(s, a) using the Eq. 3
   if not have n_i defined then
      insert n_i on Q − table
   end if
   i ← i + 1
until no more interaction
```

4.2.1 Economic TG

The Economic TG (ETG) was proposed by us to be inserted in the robotic architecture, presented in Section 3, in the learning mechanism module (Fig. 1). It is based on the works of Driessens [7], Mccallum [17] and Kearns and Mansour [13]. The works proposed by Mccallum [17] and Kearns and Mansour [13], RL algorithm are combined with a tree based method to store examples. However, they do not use relational representation.

ETG is an enhancement of TG algorithm. Due to the characteristics of the architecture in relation to the learning mechanism, some changes were necessary to be done in ETG algorithm. Basically, these are related to the necessity value produced by consequence control module (explained in Section 3.2), in which is used in the process of example storage in the tree and by the fact that ETG do not use any property of TILDE system [2].

TG algorithm interacts with the environment until find final state. After, the examples are inserted in the tree using a specific metric to insert only good examples from final state until the first state. For this a look up table is used.

In ETG algorithm, they are inserted in the tree at each interaction. This occurs because we decide by do not use a auxiliary table to store the examples and metrics before insert them into the tree. ETG algorithm learns a control policy for an agent as it moves through the environment and receives rewards for its actions. An agent perceives a state s_i, decides to take some action a_i, makes a transition from s_i to s_{i+1} and receives the reward r_i. The task of the agent is to maximize the total reward it gets while doing actions. Agents have to learn a policy which maps states into actions.

In Algorithm 2 is showed the processing of ETG. The algorithm starts by initializing the Q-function and creates an empty regression tree [34].

The learning mechanism takes from the environment state, an necessity of the agent, then it chooses (using the current policy) and takes an action. This process changes the state and the agent receives its reward. The reward can be either positive (equals to 10) or negative (equals to -1). After this occurred, the *qvalue* is computed by:

$$\hat{Q}_i \leftarrow Q(s_i, a_i) + \alpha \left[r_{i+1} + \gamma \max(Q(s_{i+1}, a_{i+1})) - Q(s_i, a_i) \right] \quad (4)$$

Then, the set of quadruples (state, action, *qvalue*, necessity) is presented to relational

regression engine. This process is repeated until there are not more interactions to be executed. All processing can be found in Algorithm 2.

Algorithm 2 ETG Algorithm

initialize the *Q-function* hypothesis $\hat{Q}_0$ and create a tree with a single leaf
$i \leftarrow 0$
repeat
  take state s_i
  take necessity n_i
  choose a_i for s_i using a policy derived from the current hypothesis $\hat{Q}_i$
  take action a_i, observe r_i and s_{i+1}
  Update $\hat{Q}_i$ using the Eq. 4
  Update relational regression algorithm using $x = (s_i, a_i, \hat{Q}_i, n_i)$ to produce $\hat{Q}_{i+1}$ {Use Algorithm 3}
  $i \leftarrow i + 1$
until forever

The relational regression engine receives a set of (state, action, *qvalue*, necessity) and tests the internal nodes if the state already exists. If this condition is false, the state is inserted in the tree and the leaf receives the action with the *qvalue* and necessity values, forming a new branch. Otherwise, it updates the *qvalue* for respective action in the leaf node.

In a leaf node, more than one action can be considered. For an easier access to the most adequate action, these actions can be ordered in decreasing order according to their *qvalue* always that an example is inserted or updated. Each leaf also has a necessity associated with action and it refers to a necessity of the robot to choose this action in that state. Here, we use only the attention necessity. A relational regression engine adopted in ETG algorithm is presented in Algorithm 3.

Algorithm 3 ETG-Regression Engine

repeat
  sort the state down the tree using the tests of the internal nodes until to reach leaf node or null
  if the node is a leaf **then**
    if action exists **then**
      the *Q-value* is updated for the action in the leaf node according to the example {time indicates to update the Q value of rule}
    else
      the *Q-value* is inserted and the necessity for the action in the leaf node according to the example {time indicates the creation of a rule}
    end if
  else if the null is attained **then**
    generate a node
  end if
until the example in a branch
if necessary **then**
  order actions in decreasing *Q-value*
end if

4.3 Analysis of Complexity of the algorithms

In this paper, we are comparing the performance of the three algorithms: Q-learning, Contingency and ETG algorithms. In this section, we will present an analysis of complexity of these algorithms.

In terms of memory requirements, the Q-learning and Contingency algorithms need the same amount of memory since they use a matrix (mxn) and a vector (mxn elements), respectively, where m denotes the number of states and n the number of actions for storing the Q values or fitness. The ETG algorithm stores all of information in a binary tree. As in a leaf node, more than one action can be considered, it is necessary a vector for saving all actions. However, this vector has a dimension limited by the number of actions, n, considered in a share attention task. So, Considering that $h > 1$ denote the depth of the binary tree, we have a total of

$$\sum_{i=0}^{h-2} 2^i + 2^{h-1}.n < \frac{1}{2(h-1)}\left[(h-1).1\right] + n < n+1$$

nodes to be stored against *mxn* memory units in both methods: Q-learning and Contingency algorithms.

In terms of memory access for recovering information, the time required for ETG algorithm

is, certainly, less than for the other algorithms. Considering that for an easier access to the most adequate action, the actions are stored in decreasing order according to their Q-value, the total time consumed is of order

$$O\left(\log_2 m\right)$$

since a binary tree structure is being used against $O(mxn)$ in the case of the matrix (in Q-learning) and vector (in Contingency algorithm).

Therefore, ETG algorithm presents the better performance than other algorithms what is shown in the experiments that will be presented in next section.

5 Experimental Results

The main results from a set of experiments carried out to evaluate our architecture are presented here. For this, we divide this section in two. The Experiment #1 is presented a comparison among the three learning methods in a sociable interaction simulator to find the most adequate to be used in real experiment and the Experiment #2 we evaluate the performance of the robotic architecture controlling a robotic head using the learning method chosen.

Before we start to talk about the two experiments performed, we introduce the metric used for analysis the three learning algorithms. The first metric measure the number of times of mutual gaze happens during the interaction. It is considered when the robot see the frontal face of human for 3 units of time. This metric is used only for simulation experiment. The other one named as *correct gaze index* (CGI) measure is used in both experiment and it is based on measures proposed by Whalen and Schreibman [40] defined as the frequency of gaze shifts from the human to the correct location where the human is looking at, given by:

$$CGI = \frac{\#\ shifts\ from\ the\ human\ to\ correct\ location}{\#\ shifts\ from\ the\ human\ to\ any\ location} \quad (5)$$

The next step, we have compared the three learning methods using a social interaction simulator.

5.1 Experiment #1

The Social Interactive Simulator (SIS) was based on works of Triesch and his colleagues [38]. In order to simulate the shared attention task, it has been defined three entities that can be manipulated through functions of the simulator. They are a human, a robot, and two toys. The human being and the robot are positioned face to face, at a distance of approximately 50 cm from each other. The simulator enables that up two toys are positioned in the social environment. A toy can be positioned at any empty place of the social environment at any moment [29].

SIS is able to simulate an interaction between a robot and a human in a controlled environment. The human being is fixed on the upper side of the interface in front of robot, that is fixed on the lower side of it. They can turn left or right their heads up to 90°. Their central focus in 0° means that they are looking for each other. When an object i is positioned in a environment, the simulator maps the angle between this object and the robot's focus, that is, the distance that the robot must move its head to focus the positioned object [29].

To quantify the learning capabilities of the architecture through the learning of gaze following, at specific points during the learning process we temporarily interrupt the learning phase to evaluate its behavior. This evaluation was done by 20 runs of 500 time units (500 s in the simulator). For each run, the CGI value, given by Eq. 5 was computed and after the 20 runs a mean and standard deviation were calculated. After the evaluation phase, the learning process was resumed. During the evaluation phase, the human initially kept the focus on the robot until it establishes eye contact with the human, characterized by 3 time units keeping eye contact. Then two objects were positioned in the environment and the human turned his gaze for one of these objects. However, in the evaluation phase, the object to which the human should turn his gaze was place on a position given by pre-established sequence (to prevent non

determinism in the results). The second object (the distract object) was placed on an empty position, obeying the probabilities defined in the social interactive simulator. Once the robot turns its head to any direction, the simulator verifies if it is looking to the correct position in the environment (a toy which the human is looking for) or not, and updates the CGI measure. This procedure takes 1 time unit. Afterward, the objects are then removed from the environment and the human turns his gaze to the robot, keeping the robot make eye contact again.

When we are dealing with shared attention, a fact very important that it must be considered in all interactions is the number of times that the robot establishes eye contact with human. This is an essential fact for shared attention. The robot, through the simulator, can one of both options: to find anything in the environment or pay attention to human. If the robot choose only the first option, it could not simulate the shared attention. Because this, it is important that the learning algorithm maximizes the number of established eye contact. Figure 4 shows the number of times that the robot establishes eye contact with human by using each one of the algorithms. In this figure, it is showed the beginning of the interaction between human and robot, in a total of 125 possible opportunities to establish eye contact each other.

Figure 5 shows the learning progress over the time. It plots the CGI measured for each evaluation phase, at specific points during the learning process. This figure shows the performance of three different learning algorithms used as a learning mechanism in the proposed architecture. In this figure, the lines presents the evaluation for different algorithms. At each 500 time units, the learning process was temporarily interrupted and an evaluation of the robot's behavior was done by 20 executions of 500 time units, we call this interval by run. In this figure, it is showed the average value the CGI measure and the standard deviation (indicated by error bars), for each run.

Initially, it can be seen that all of the algorithms have not any knowledge about the problem. After the first run, all algorithms improve the robot knowledge attaining near of 80% of maximum CGI value. In this stage, the robot or agent learns a lot about the problem. After this, the contingency learning do not get to improve its knowledge until the end and this fact can be seen by your curve, remaining constant. ETG and *Q-learning* algorithms improve their learning. Their curves are increasing over the time until to attain a stabilization level.

A deeper analysis can be made considering Figs. 4 and 5. Considering the factors of learning and the number of established eye contact, one can say that ETG algorithm achieved better results. The experimental results showed that the contingency learning algorithm established a greater number of eye contact than others, but it

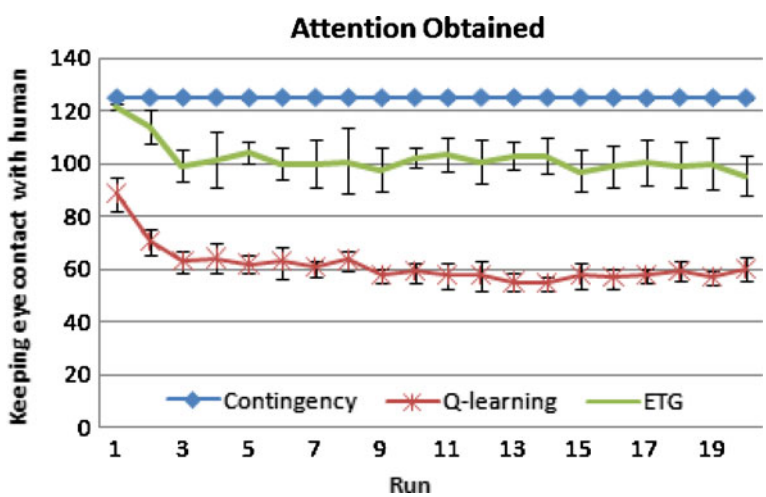

Fig. 4 Capacity of attention

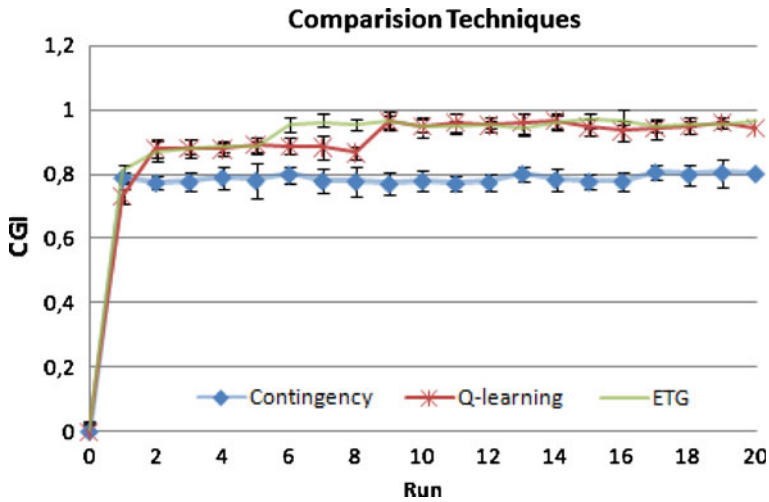

Fig. 5 Learning evolution during the experiments

did not get a good CGI value compared to others. On the other hand, the *Q-Learning* established a lower number of eye contact than others, but it got a good CGI value. Overall, ETG presented a average value better in both experiments.

Another factor analysis is in relation to the memory requirements necessary for the algorithms: *Q-Learning* and ETG. *Q-Learning* algorithm uses as a mechanism for knowledge representation, in this case, a *Q-Table* of 3,150 positions of memory. ETG algorithm used 68 nodes, with 29 leaf nodes in which 328 positions were occupied for the storage of actions, totalizing 396 positions of memory. This demonstrates clearly that ETG algorithm required less memory than the *Q-Learning* algorithm.

Then, ETG algorithm was chosen to be used in the real experiment with robotic head. To follow, it will be presented a second experiment that has been carried out by one of the authors (caregiver), employing the interactive robotic head previously presented.

5.2 Experiment #2

The purpose of this experiment was to evaluate the capabilities of the proposed architecture on exhibit appropriate sociable behavior and learning in a real and controlled environment. In the experimental scenario, a caregiver established eye contact with the robot and then presented three different objects (fruits represented by an apple, lemon and orange) in order to teach the robot for following his gaze and after to do a declarative pointing of it.

For this, four stimuli have been declared: *face*, *object*, *attention*, and *environment*, in which attention was configured as reinforcer stimuli. Three facts have been declared to define that red, orange and green objects are fruits. Six facts have been declared in order to differentiate the human's head pose: in frontal pose, two poses of left profile, two poses of right profile and one pose of down profile. Additionally, two more facts have been declared to define when the robot is focusing the human or a fruit.

The response emission module was configured as it follows. The constant learning (α parameter) has been set with a value equal to 0.2. The discount factor (γ parameter) was set to 0.9. The exploration factor(ϵ parameter) was set to 0.05. Seven responses have been defined so that the robot could look to the humans or search for a fruit in the five defined regions (down, left, right, left down and right down) by turning its head to the left or right side. This was done to divide the environment in areas of interest that would turn possible for the robot to learn following the gaze of a human being to correct places.

The motivational system was configured as it follows. The necessity unit was created: socialize. The activation threshold of the motivational

system has been set to 0.50. The sigmoid function inclination (δ parameter) has been set with a value equal to 0.20. For the necessity unit socialize, the bias has been set with value equal to 1.00 and weight of its connection has been set with a value equal to 0.5. The weight of the recurrent connection has been set with a value equal to 1.00. The weights of the connections of the input units (hear (attention), see (frontal (face)), see (looking_frontal (face)), see (looking_fruit (object))) have been set, respectively, with the values 1.50, 0.95, 0.50 and 1.50. All the constants have been chosen empirically.

The experiments were composed by learning phase of 100 time units or runs. During the learning phase, the human being initially kept the focus on the robot until it establishes eye contact with him, characterized by looking each other. Then, one fruit were positioned in the environment and the caregiver turned his gaze for this fruit. Afterwards, the robot does a declarative pointing with safety or uncertainty about a particular fruit. Finally, the fruit is then removed from the environment and the human turns his gaze to the robot, keeping the eye contact with the robot again. This procedure is done in order to simulate an interaction where two agents are keeping eye contact and then one turns his gaze to an interesting event or object.

In the first 30 time units of learning phase, no objects are positioned in the environment and the caregiver kept his focus on the robot the whole time, so the robot has learned to obtain the caregivert's attention by keeping eye contact with him, satisfying its necessity of *socialization*. This procedure was done to shape the robot's behavior of looking for a caregiver and keeping eye contact whenever it feels necessity of socialization. After this, the learning phase was resumed using a fruit as stated above. During the learning, the robot looks for him whenever it wants to socialize. However, when an object is positioned in the environment and the caregiver turns his gaze to it, the robot looses his attention and starts to seek anything in the environment that can satisfy its internal states of *socialization*. Additionally, if the robot looks for a fruit which the caregiver is keeping his gaze, the person returns his attention to the robot, in relation to the fruit. In this way, after a history of reinforcement, the robot will learn to follow the caregiver's gaze to receive his attention and to satisfy its needs of socializing.

The learning capabilities of the architecture was analyzed by observing the robot interacting with the caregiver and the environment, and computing a CGI measure. To quantify the learning capabilities of the architecture through the learning of gaze following, at specific points during the learning process, we the temporarily interrupt the learning phase to evaluate its behavior. This evaluation was done by 5 runs of 100 time units. For each run, the CGI value, given by Eq. 5 was computed and after the 5 runs a mean and standard deviation were calculated. After the evaluation phase, the learning process was resumed. During the evaluation phase, the human initially kept the focus on the robot until it establishes eye contact with the human. Then a fruit was positioned in the environment and the human turned his gaze for one of these objects.

However, in the evaluation phase, the object to which the human should turn his gaze was place on a position given by pre-established sequence (to prevent non determinism in the results). Once the robot turns its head to any direction, the software in robotic head verifies if it is looking for the correct position in the environment or not, and update the CGI measure. Afterwards, the objects are then removed from the environment and the human turns his gaze to the robot, keeping eye contact with robot again.

To test the ability of the robotic head to do a declarative pointing step, it was taught for it how to recognize fruits, such as apple, lemon and orange. For the learning of the fruits, it was used a neural network ART2 [10] to recognize different colors. The ART2 network is a good technique for clustering data. It was evaluated this neural network with five possibilities: Unknown, Correct guess, Incorrect guess, Error or Success. At first time, when the robot sees a new fruit, the caregiver tells its name for the robot. After, the robot tries to express the name of the fruit doing the association between name of fruit and its color. The presentation phase was repeated five times (five executions) and we did not change the light conditions during the experiments. Then, after five executions, the average value and standard

deviation, for each measure in the five executions, have been calculated. The neural network ART2 was successful to do this task classifying correctly 73% of the patterns trained.

In Fig. 6, it is presented the curve that demonstrates the progress of the learning during the experiments. This figure shows a chart that presents the value of the average of the CGI values, in specific points during the learning phase. The obtained results show that CGI values increases over the learning phase, demonstrating the learning capacities of the architecture with ETG algorithm.

A great increasing in a learning curve is noted in the beginning until the second running when the robotic head builds its first knowledge. After this, based on all new facts that happened in the environment, it adjusts its knowledge base. This process is shown in low increasing the curve. At the certain time, when the curve becomes fixed, or it has a little change, for a time period, the robot has the optimal reply for this problem.

In Table 1 is showed the average values and standard deviation of the five measures for the five executions carried out during the experiments [33].

The robotic head shows be able to associate visual and auditory stimulus if we analyze the results of the declarative pointing step. With the good performance to identify a fruit, the architecture was able to learning with a caregiver's tutelage. The complete result of the declarative pointing step can be seen at [33].

These experiments was carried out by the modeling of the behavior of looking for a human, followed by the behavior of follow the human's gaze

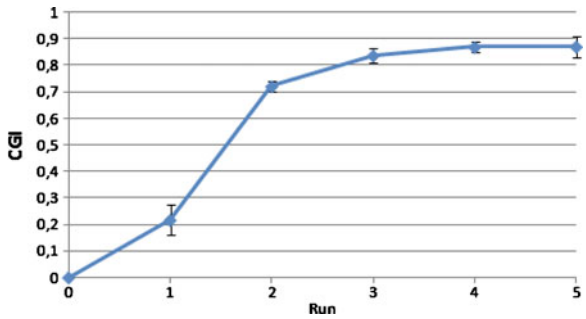

Fig. 6 Learning evolution during the experiments

Table 1 Results obtained after the five executions of guided learning sessions [33]

Measure	Average rate (%)
Unknown rate	7.23 ± 0.58
Correct guess rate	17.47 ± 1.28
Incorrect guess rate	1.8 ± 0.44
Error rate	0.6 ± 0.2
Success rate	72.89 ± 2.19

and, finally, to perform the declarative pointing step. The results show that the architecture is able to exhibit appropriate behaviors during a real and controlled social interaction. Additionally, the results show that the proposed architecture is able to acquire basic sociable abilities from existing innate behaviors in the repertoire of the robot and also through the interaction with the environment. The results also evidence that the architecture constitutes a contribution for the research area of shared attention emergence, representing a computational model able to simulate the learning of this hard sociable skill.

6 Conclusions

In this paper, it was presented a robotic architecture inspired on Behavior Analysis. Three different learning methods, contingency learning, Q-learning and ETG algorithm, were compared aiming to provide to the robotic head the ability to perform the declarative pointing step by simulation. An analysis of the computational resources necessary for the three algorithms was done indicating that ETG algorithm has a better performance than others. So, ETG was incorporated to the architecture learning mechanism. The results showed that the architecture was able to exhibit appropriate behaviors during a real and controlled social interaction, through of learning from sociable interactions, that is, with caregiver's tutelage.

Future works include the extension of the architecture by implementing new mechanisms and skills like *emotion*, *verbal behavior*, *long term interaction control* and *learning by imitation*.

Acknowledgements The authors would like to thank FAPESP, CNPq and CAPES for all support received.

References

1. Animated Head System With Camera: Servo & Multimedia Controller Board WHA8030, Dr. Robot
2. Blockeel, H., De Raedt L.: Top-down induction of first-order logical decision trees. Artif. Intell. **101**, 285–297 (1998)
3. Breazeal, C.: Designing Sociable Robots. MIT Press, Cambridge (2002)
4. Breazeal, C., Scassellati, B.: A context-dependent attention system for a social robot. In: International Joint Conference on Artificial Intelligence, pp. 1146–1153 (1999)
5. Catania, A.C.: Learning, Interim 4th edn. Sloan, Santa Rosa (2006)
6. Cooper, J.O., Heron, T.E., Heward, W.L.: Applied Behavior Analysis, 2nd edn., 0131421131. Prentice Hall, Englewood Cliffs (2007)
7. Driessens, K.: Relational Reinforcement Learning. Katholieke Universiteit Leuven, Leuven (2004)
8. Gadanho, S.C., Hallam, J.: Robot learning driven by emotions. Adapt. Behav. **9**, 42–64 (2001)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman, Boston (1989)
10. Haykin, S.: Neural Networks—A Comprehensive Foundation. Prentice Hall, Englewood Cliffs (1999)
11. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. Pattern Anal. Mach. Intell. **20**, 1254–1259 (1998)
12. Kaplan, F., Hafner, V.: The Challenges of Joint Attention. Lund Univ. Cogn. Stud. **117**, 67–74 (2004)
13. Kearns, M., Mansour, Y.: On the boosting ability of top-down decision tree learning algorithms. In: STOC '96: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, pp. 459–468 (1996)
14. Kim, H., Jasso, H., Deak, G., Triesch, J.: A robotic model of the development of gaze following. In: 7th IEEE International Conference on Development and Learning (ICDL 2008), pp. 238–243 (2008)
15. Lti Lib: Lti Image Processing Library—Developerś Guide. Lehrstuhl fuer Technische Informatik - Aachen University of Technology (2003)
16. Matsuda, G., Omori, T.: Learning of joint visual attention by reinforcement learning. In: Int. Conf. on Cognitive Modeling (ICCM), pp. 157–162 (2001)
17. Mccallum, A.K.: Learning visual routines with reinforcement learning. In: AAAI Fall Symposium, pp. 82–86 (1996)
18. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
19. Morency, L-P., Sundberg, P., Darrel, T.: Pose estimation using 3d view-based eigenspaces. In: Analysis and Modeling of Faces and Gestures (AMFG'03), pp. 45–52 (2003)
20. Morik, K., Wrobel, S., Kietz, J-U., Emde, W.: Knowledge Acquisition and Machine Learning: Theory, Methods, and Applications. Academic, San Francisco (1993)
21. Mutlu, B., Hodgins, J.K., Forlizzi, J.: A storytelling robot: modeling and evaluation of human-like gaze behavior. In: Proceedings of HUMANOIDS'06, 2006 IEEE-RAS International Conference on Humanoid Robots, pp. 518–523 (2006)
22. Mutlu, B., Shiwa, T., Kanda, T., Ishiguro, H., Hagita, N.: Footing in human-robot conversations: how robots might shape participant roles using gaze cues. In: HRI '09: Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction, pp. 61–68 (2009)
23. Nagai, Y.: The role of motion information in learning human-robot joint attention. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation(ICRA), pp. 2069–2074 (2005)
24. Nagai, Y., Hosoda, A., Asada, M.: A constructive model for the development of joint attention. Connect. Sci. **15**, 211–229 (2003)
25. Nuance: Introduction to the Nuance System. Nuance Communications Inc., South Yarra (2001)
26. Otterlo, V.: A Survey of Reinforcement Learning in Relational Domains. CTIT Technical Report, TR-CTIT-05-31 (2005)
27. Pierce, W.D., Cheney, C.D.: Learning, Interim 4th edn., 0805862609. Psychology Press, Hove (2008)
28. Policastro, C.A.: Arquitetura robótica inspirada na análise do comportamento. PhD thesis, Universidade de São Paulo (2008)
29. Policastro, C.A., Romero, R.A.F., Zuliani, G., Pizzolato, E.: Learning of shared attention in sociable robotics. J. Algorithms **64**, 139–151 (2009)
30. Scassellati, B.: Mechanisms of shared attention for a humanoid robot. pp. 102-106 (1996)
31. Scassellati, B.: Imitation and Mechanisms of Joint Attention: A Developmental Structure for Building Social Skills on a Humanoid Robot, pp. 176–195. Springer, New York (1999)
32. Shon, A.P., Grimes, D.B., Baker, C.L., Hoffman, M.W., Zhou, S., Rao, R.P.N.: Probabilistic gaze imitation and saliency learning in a robotic head. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), pp. 2865–2870 (2005)
33. Silva, R.R., Policastro, C.A., Zuliani, G., Pizzolato, E., Romero, R.A.F.: Concept learning by human tutelage for social robots. Learning and Nonlinear Models **6**, 44–67 (2008)
34. Silva, R.R., Policastro, C.A., Romero, R.A.F.: Relational reinforcement learning applied to shared attention. In: Proceedings of 2009 International Joint Conference on Neural Networks, pp. 2943–2949 (2009)
35. Sumioka, H., Yoshikawa, Y., Asada, M.: Reproducing interaction contingency toward open-ended development of social actions: case study on joint attention. IEEE Trans. Auton. Mental Develop **2**, 40–50 (2010)
36. Staudte, M., Crocker, M.W.: Visual attention in spoken human-robot interaction. In: HRI '09: Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction, pp. 77–84 (2009)

37. Tan, P-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley Longman, Boston (2005)
38. Triesch, J., Teuscher, C., Deak, G.O. Carlson, E.: Gaze following: why (not) learn it?. Dev. Sci. **9**, 125–147 (2006)
39. Watkins, C.J.C.H.: Learning from Delayed Rewards. University of Cambridge, Cambridge (1989)
40. Whalen, C., Schreibman, L.: Joint attention training for children with autism using behavior modification procedures. J. Child Psychol. Psychiatry **44**, 456–468 (2003)
41. Yamazaki, A., Yamazaki, K., Kuno, Y., Burdelski, M., Kawashima, M., Kuzuoka, H.: Precision timing in human-robot interaction: coordination of head movement and utterance. In: CHI '08: Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems, 978-1-60558-011-1, pp. 131–140 (2008)
42. Yu, C., Scheutz M., Schermerhorn, P.: Investigating multimodal real-time patterns of joint attention in an HRI word learning task. In: HRI '10: Proceeding of the 5th ACM/IEEE International Conference on Human–Robot interaction, pp. 309–316 (2010)