2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

Robot Learning by Observation based on Bayesian Networks and Game Pattern Graphs for Human-Robot Game Interactions

Hyunglae Lee, Hyoungnyoun Kim, Kyung-Hwa Park, and Ji-Hyung Park
Intelligence and Interaction Research Center
Korea Institute of Science and Technology, Seoul, Korea
{horry, nyoun, kh_park, jhpark@kist.re.kr}

Abstract—**This paper describes a new learning by observation algorithm based on Bayesian networks and game pattern graphs. Even with minimal knowledge of a game or human instructions, the robot can learn the game rules by watching human demonstrators repeatedly play the game multiple times. Based on the knowledge acquired from this learning process, represented in Bayesian networks and game pattern graphs, the robot can play games as robustly as humans do. Our learning algorithm for human-robot game interaction is implemented using a teddy bear-like robot and is demonstrated by application to well-known social games, specifically Rock-Paper-Scissors, Muk-Chi-ba and Blackjack.**

I. Introduction

In recent years human-friendly robots have been receiving much attention in both academia and industry [1], and several entertainment robots adopting cutting edge artificial intelligence techniques and dynamics technology - for example, Sony's AIBO and WowWee's ROBOSAPIEN – have come onto the market. In addition, there many efforts have been made to develop intelligent game-playing robots such as chess-playing robots [2] and various soccer-playing robots [3]. However, almost all of the behaviors of the above-mentioned robots operate according to formal rules or are executed in a predefined manner.

Several game-playing robots with learning abilities have been suggested in recent years. A framework that enables robots to learn from observations using task primitives was proposed [4] and applied to air hockey and a marble maze game [5]. Although this framework based on primitives provides flexibility to game learning and enables autonomous game playing, the learning focuses on simple tasks, rather than complex game interaction rules. Leonardo [6], one of the most socially intelligent and cooperative robots, showed the social ability to entertain its human opponent while learning and playing a button pressing game [7]. Considering famous social games that people, especially children, usually play, Leonardo still faces limitations coping with more complex and subtle games. Recently, research on learning interaction rules based on an imitation faculty with a simple visuo-motor mapping was suggested [8]. However, the suggested method for acquiring the interaction rules is too simple to be used in the game domain.

In addition, although much work has been done on game learning in the field of artificial intelligence (AI), most of it has concentrated on a single game, such as chess [9, 10] or poker [11], rather than learning general game playing. Thus, even the best chess-playing programs or robots cannot play simple card games. CogVis [12] project is another famous related work and similar to our research in that it focuses on learning by observation and deals with the Rock-Paper-Scissors game. Even though it demonstrated on the Rock-Paper-Scissors game based on symbolic protocol learning, no further game examples are provided to verify the validity of the algorithm and its scalability. In addition, the human played game with a virtual player not with a real robot, which lacks interactivity between the human and robot.

In the present work we propose a game learning algorithm for natural human-robot game interaction based on Bayesian Networks (BNs) and game pattern graphs. These algorithms enable the robot to learn game interaction rules automatically by repeatedly observing several real game plays performed by human demonstrators. After learning the rules, the robot knows what to do when it encounters certain situations while playing the game. In addition, the robot plays games with human players robustly even when it encounters unusual and unforeseen events. Experiments on three famous social games verify the suggested learning algorithms and show natural human-robot game interaction.

In the following section we describe the game domain applied to our human-robot game interaction. In Section III, we explain the details of the game learning algorithms. Section IV describes the method's implementation and presents the experimental results of the method's application to three kinds of social games. Finally, Section V and VI present conclusions and future work respectively.

II. Game Domain

A game consists of several patterns, and progresses with the repetition of pattern changes. The pattern of the next step is determined by the actions of players at current step, or those of current and previous steps. For example, the next pattern in Rock-Paper-Scissors is governed only by the

This research was supported by the Devlopment of Cooperative Network-based Humanoid Technologies Project of the Ministry of Information and Communication (MIC) of Korea.

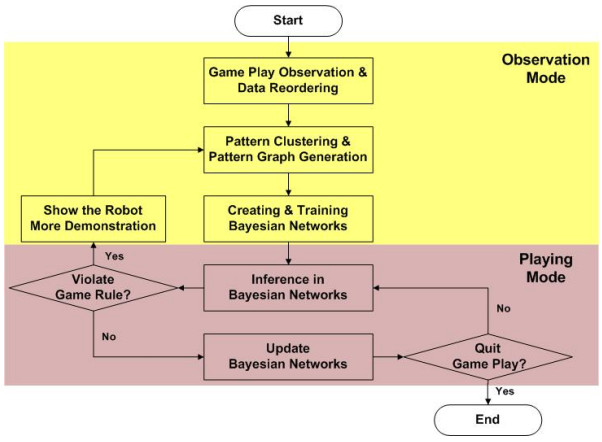


Fig. 1. Overall process of game learning by observation and game playing

players' current action states; whereas in Poker, the actions of both previous and current steps affect the next pattern or, in other words, players' next actions.

In this research we focus on simple games that even children can easily learn and play, and we consider only games in which the current step's actions determine the next step. In addition, to emphasize the human-robot interaction aspect, we focus on interactive games that usually involve face-to-face interaction with sounds, motions, and visual information during game plays. According to this domain, we selected Rock-Paper-Scissors, one of the most famous social games, and Muk-Chi-Ba, a variant of the Rock-Paper-Scissors originating in Korea, as subjects for robot game learning. We also tested on three-player Rock-Paper-Scissors and the famous card game Blackjack to verify the suggested game learning algorithm and to show its scalability.

III. Game Learning Algorithms

Each game progresses with a repetition of pattern changes. Thus, learning the game rules means knowing the causes and timing of pattern changes, knowing how to react to them, and finally, knowing how to win the game. Our robot has the ability to learn various social games. First the robot observes human demonstrators' game plays several times and stores all the observed data, based on which the robot generates the game rules with several sequential operations. After creating these rules the robot plays the game with human players by following the generated rules. The overall process of game learning is represented in Fig. 1, and the details of each operation are described below.

A. Game Observation

When the robot observes humans playing the game, it recognizes the players' actions (sounds, motions) and visual information, if it exists, and stores them in order of time (top left of Fig. 2). To use these data in a learning process, actions or vision information that co-occur are grouped together and

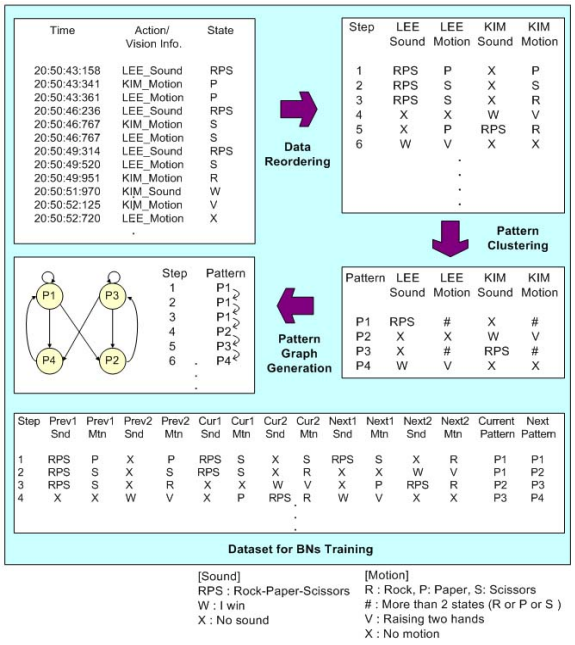


Fig. 2. An example from data reordering to pattern graph generation and a dataset for BNs training (a two-player Rock-Paper-Scissors game: consists of sound and motion data without visual information)

the game data are reorganized according to the grouping information (top right of Fig. 2). Here we call each group a step of the game. Considering that at least a one-second interval exists between adjacent steps, such as turn-taking in a game, we use one second as the minimum interval for separating steps. We can adjust this value according to the type of game.

After reordering the game data we can break the game down into several patterns by scanning from the first step to the last step of the game process, where the existence or nonexistence of each player's sound, motion and vision states is a key criterion for the pattern clustering (middle right of Fig. 2). Pseudocode for the pattern clustering is described below.

```
FOR each step of the game process
  IF the current step belongs to the existing pattern set THEN
    Do nothing
  ELSE
    IF the current step and one of the existing patterns can be a
    subset of a newly created pattern THEN
      The newly created pattern replaces the selected existing
      pattern
    ELSE
      Add the current step to the pattern set as a new pattern
    END IF
  END IF
END FOR
FOR each pattern
  Calculate the correlation among each player's action states
  (sound and motion)
  IF the pattern has different correlation properties among
```

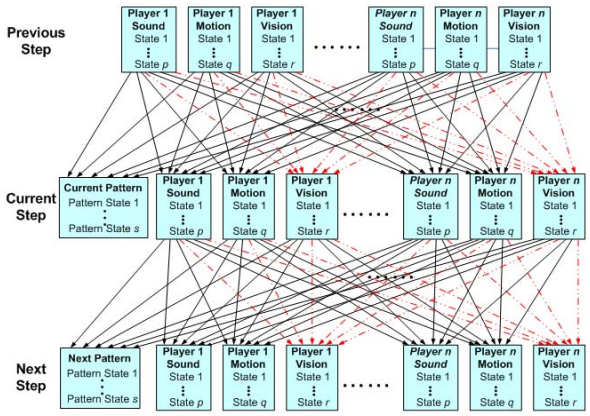


Fig. 3. Game model represented as Bayesian networks

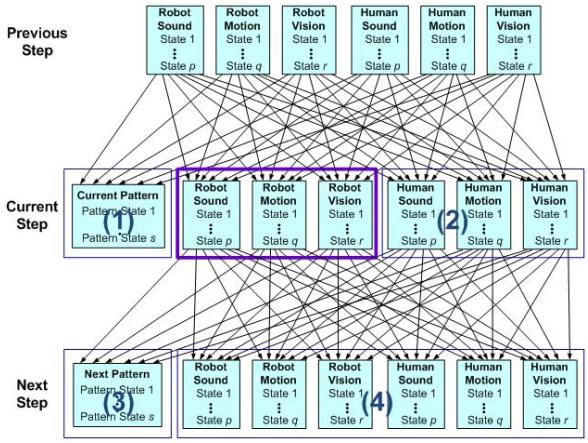


Fig. 4. Probabilistic inference procedure in game playing based on the trained Bayesian networks

action states THEN
Subdivide the pattern into several patterns
END IF
END FOR

Several examples of pattern clustering are given below. At Step 2 of Fig. 2, a newly created pattern replaces the existing pattern because the existing pattern "RPS / P / X / P" and the current step "RPS / S / X / S" can be a subset of new pattern "RPS / # / X / #". Here, "#" means that more than two states can be the candidates for this action state. At Step 4, the current step "X / X / W / V" is added to the pattern set because there are no special inclusive relationships between the current step and existing patterns.

In addition, the correlation value among each player's action states is a criterion for subdividing the pattern. For instance, consider the pattern "# / # / X / #," where the first and second "#" have four and three states, respectively. If one state (A) of the first "#" is weakly correlated with all states of the second "#" and the other states of the first "#" are strongly correlated with one of states of the second "#" (i.e. correlation value > 0.8), we can subdivide the pattern "# / # / X / #" into "A / # / X / #" and "#-A / # / X / #". Here, "#-A" means that more than two states except "A" can be the candidates for this action state.

After the pattern clustering operation, each of the game steps corresponds to the appropriate game pattern. A game pattern graph, which provides essential information on the game playing process, is automatically generated in the form of a digraph (middle left of Fig. 2). Each of the game patterns becomes a node of the pattern graph, and edges are created by connecting each node of the previous step with a node of the current step consecutively. Because robust pattern clustering is important in the game learning process, abnormal steps containing ambiguous action or visual data are excluded from both in pattern clustering and in pattern graph generation.

In the games that we specify in Section II, one player's current actions (sound and motion) and visual information may affect not only his or her next actions and visual information depending on the type of game but also those of other players. In addition, the next pattern is determined according to all players' current actions and visual information. We can clearly represent these relations using BNs [13], as shown in Fig. 3. Each of the BNs' nodes is a variable representing each player's actions or visual information, or a game pattern at a certain moment, and each node can have various states. Edges represent the direct dependencies between nodes. Dotted red lines mean that edges between nodes can be created or not, depending on the type of game. If the game consists of changes of vision information besides of players' sound and motion, we can create incoming edges on vision nodes, and otherwise not. Thus, BNs can fully represent the casual relationships over three steps, i.e., the previous, current and next steps, and this structure is used to describe games that belong to the game domain explained in Section II.

Given this structure, conditional probability distributions between nodes can be determined by training the BNs' parameters. We can train the BNs using the dataset obtained from the data reordering and the pattern clustering processes. An example of the training dataset is given at the bottom of Fig. 2. Each data in the dataset becomes the finding of each node in BNs. After successfully completing the training process, the robot stores the trained BNs together with the game pattern graph, and utilizes this information in game playing.

B. Game Playing

When playing games with human players, the robot first transforms its viewpoint to one of the human players in the BNs. For example, in Fig. 3, "Player 1 Sound," "Player 1 Motion" and "Player 1 Vision" are replaced with "Robot Sound," "Robot Motion" and "Robot Vision." After this viewpoint transformation, the robot performs probabilistic inference in the BNs to decide which action to execute at the

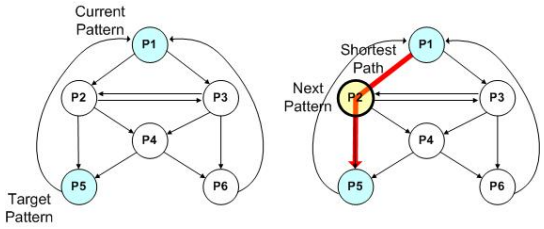


Fig. 5. Next pattern estimation in the pattern graph using the Dijkstra algorithm

current moment. This probabilistic inference procedure is explained below with Fig. 4. Although the explanation is based on a two-player game, it also can be applied to games with more than two players.

The first step in probabilistic inference is to decide the pattern state at current step according to the results (findings of nodes in BNs) of the previous step ((1) in Fig. 4). The state with the highest probability becomes the finding of the "Current Pattern" node.

After setting "Current Pattern" node's state, the robot examines its current action nodes' probability distributions ("Robot Sound," "Robot Motion" and "Robot Vision," if they exist). For each node, if the probability value of one state is much bigger than others, the robot can easily decide current actions to execute without further inferences. In this work, the robot sets current action states if the highest probability value is more than five times the second highest value.

Otherwise, the robot determines its current actions through sequential operations from (2) to (4). First, it estimates the human player's current actions (2). The probabilities of the current action states are influenced by the previous step's findings. States with the highest probability become the estimated findings of the human player's current actions.

In general, the game player's goal is to win the game. In this work, we assumed that the robot knows the winning state that raising two arms with shouting "I Win" means winning the game. To achieve this goal, the robot should try to change the pattern state in the direction of a target state, i.e., the winning state. The robot solves the shortest path problem from the current pattern state to the target pattern state based on the Dijkstra algorithm [14]. When the shortest path is calculated, the state of the next pattern is determined automatically (3). Fig. 5 shows an example of this process.

In addition, when the next pattern state is determined, several of the next step's nodes may be fixed accordingly, because some game patterns have several fixed action states. For example, pattern P2 in Fig. 2 always has a fixed state of "X / X / W / V."

Based on the determined node information of BNs, explained above, the robot can select the actions to execute at the current step. The states with the highest probability of each action node become the current actions to execute.

After the robot and the human player have executed their actions at the current step, all nodes in the BNs are released to their initial states and the BNs are updated for the next-step's interaction. In detail, all players' current actions, as well as their data from the previous two steps, become the new findings for updating the BNs. Thus, when playing the game with a human player, the robot gradually learns the human player's play pattern, enabling it to adaptively react by utilizing the BNs' update information.

Fig. 6. Motion primitives used in games ("Initial position, " "Rock, " "Paper, " "Scissors" and "I Win" from left to right)

Fig. 7. Hardware used in the human-robot game interactions

In addition, when the robot makes a wrong inference (i.e., violates a game rule), the human player stops the game play and demonstrates the rule that the robot missed. After these demonstrations, the human player and the robot can resume the game.

IV. Implementation & Experiments

A. Implementation

Our experimental platform for the human-robot game interaction is a teddy bear-like robot called TeddyRobi. The robot is simply designed using several servomotors from Robotis [15]. It is approximately 40cm tall and has 19 degrees of freedom (DoF): one 1 DoF neck, two 3 DoF arms, two 1 DoF hands and the remainder for legs, waist and hips. Although not very accurate, the robot is able to generate all of the motions required for the game interaction, such as moving its head, arms and legs in a manner similar to that of a human body.

The robot can gather auditory information through microphones installed on the bodies of human players. The robot recognizes the speech of humans using a speech recognition engine and responds with speech generated by a TTS (text-to-speech) engine both from Voiceware [16]. In addition, the robot can identify each person's ID using speaker recognition algorithms based on Gaussian mixture models (GMMs) [17]. Before game interactions, we recorded the voice of game players to create GMMs and trained them with the recorded data. When learning or playing game, the robot recognizes the speech of the players by comparing input

Fig. 8. Experiments of human-robot game interaction

data with trained models.

Similarly, the robot can recognize motions of human players. To simplify the game interaction a set of motions, such as "Rock," "Paper," "Scissors," "I win," and "Initial Position", to be used in game interactions are predefined as motion primitives (see Fig. 6). In addition, for efficient motion recognition, Rock, Paper, and Scissors gestures are made with the entire arms, not with the more usual hand positions (i.e. flat hand for paper, fist for rock, and scissor-shaped fingers for scissors), and robot motions corresponding to each motion primitive are created manually. Humans with a 3-axis accelerometer mounted on each wrist performed the postural changes of each motion primitive, and we extracted acceleration data (six values: two x / y / z axis value). Using the extracted data, we created Hidden Markov Models (HMMs) [18] and trained them. During game interactions, the robot recognizes opponents' motions by comparing sensed acceleration data with stored trained models.

All perceptual inputs are transmitted to the control PC via Zigbee wireless communication and the processed information that controls the robot is fed back to the robot with minimal latency. Fig. 7 shows the robot TeddyRobi, a microphone, and accelerometers used in the game interaction.

B. *Experiments*

To verify the proposed game learning algorithms, we conducted experiments on Two-player Rock-Paper-Scissors and Muk-Chi-Ba in the real environment. In addition, Three-player Rock-Paper-Scissors and Blackjack (also known as Twenty-one) were simulated to verify and show the scalability of the suggested algorithm.

Five people, three males and two females, participated in the experiments. As a pre-process of game interactions, we recorded each participant's voice for one minute to create speaker models, and constructed human motion models based on each participant's gesture data for each motion primitive. We also evaluated the accuracy of speech, speaker and motion recognition. Each human participant uttered the five kinds of words and acted out the five kinds of motions used in games. This process was repeated five times for each human participant. The accuracy rate of speech, speaker and motion recognition was 99.2%, 100.0%, and 98.4%, respectively, which is sufficient for the system to be used in human-robot game interactions.

In each experiment, the robot observed game plays

Pattern	Player 1 Sound	Player 1 Motion	Player 2 Sound	Player 2 Motion	Action Combination
P1	RPS	# {R, P, S}	X	# {R, P, S}	9
P2	X	# {R, P, S}	RPS	# {R, P, S}	9
P3	W	V	X	X	1
P4	X	X	W	V	1
	[Sound] RPS : Rock-Paper-Scissors W : I Win X : No Sound		[Motion] R : Rock, P : Paper, S : Scissors V : Raising two hands X : No motion		nAC = 20

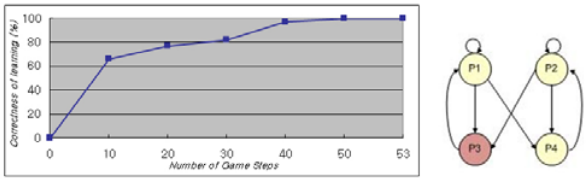


Fig. 9. Results of the Two-player Rock-Paper-Scissors game: Results of pattern clustering (top), Graph of correctness of learning vs. Number of observed game steps (bottom left), Game pattern graph (bottom right)

between human demonstrators for a few minutes and saved the game process. After observing the game being played, the robot runs the learning process. To analyze the effect of the number of observed game steps in learning process, the correctness of game learning is calculated according to the number of game steps. The correctness of game learning is defined as (1).

$$Correctness\ of\ \ game\ learning\,(\%) = \frac{nCR}{nAC} \times 100 \qquad (1)$$

nAC = number of possible action combinations in a game play
nCR = number of correct responses to all possible action combinations after the learning process

Here, correct response is a response that does not violate the game rule. When the correctness of game learning value reaches 100, we can say the robot learned the game rules perfectly. In addition, the validity of the created game pattern graph is compared with the actual game rules.

After learning the game rules, the robot played games with a human, and the performance of the robot using the learned game rules was evaluated. Finally, to test the robustness of game interaction, several unusual events were given to the robot during game playing. The human–robot game interactions during the experiments are shown in Fig. 8 and in the accompanying video clip.

1) GAME 1: Two-player Rock-Paper-Scissors game

This game starts by shouting the name of the game "Rock Paper Scissors". On the count Scissors, the players move their arms into one of three gestures: Rock, Paper, Scissors. The winner is determined by the following rules: Rock beats Scissors, Paper beats Rock, and Scissors beats Paper. If both players choose the same gesture, the game is tied and played again. As explained above, Rock, Paper, and Scissors gestures are made with the entire arms instead of usual hand

Pattern	Player 1 Sound	Player 1 Motion	Player 2 Sound	Player 2 Motion	Action Combination
P1	RPS	# {R, P, S}	X	# {R, P, S}	9
P2	#-RPS {R, P, S}	# {R, P, S}	X	# {R, P, S}	9
P3	X	# {R, P, S}	RPS	# {R, P, S}	9
P4	X	# {R, P, S}	#-RPS {R, P, S}	# {R, P, S}	9
P5	W	V	X	X	1
P6	X	X	W	V	1
	[Sound] RPS : Rock-Paper-Scissors R : Rock, P : Paper, S : Scissors W : I Win X : No Sound		[Motion] R : Rock, P : Paper, S : Scissors V : Raising two hands X : No motion		*nAC* = 38

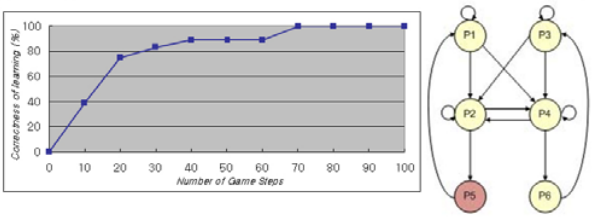


Fig. 10. Results of the Muk-Chi-Ba game

Pattern	Player 1 Sound	Player 1 Motion	Player 2 Sound	Player 2 Motion	Player 3 Sound	Player 3 Motion	Action Combination
P1	RPS	# {R, P, S}	RPS	# {R, P, S}	RPS	# {R, P, S}	27
P2	RPS	# {R, P, S}	RPS	# {R, P, S}	X	X	9
P3	X	X	RPS	# {R, P, S}	RPS	# {R, P, S}	9
P4	RPS	# {R, P, S}	X	X	RPS	# {R, P, S}	9
P5	W	V	X	X	X	X	1
P6	X	X	W	V	X	X	1
P7	X	X	X	X	W	V	1
	[Sound] RPS : Rock-Paper-Scissors W : I Win X : No Sound			[Motion] R : Rock, P : Paper, S : Scissors V : Raising two hands X : No motion			*nAC* = 57

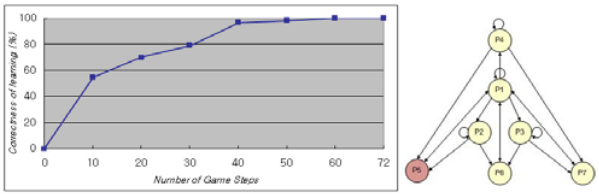


Fig. 11. Simulation results of the Three-player Rock-Paper-Scissors game

positions.

The robot observed the game demonstration for 3 minutes, extracting 53 game steps. In the learning process, 2×53 steps were used because same game rules can be applied to both players. The correctness of the game learning increased with the number of observed game steps. After acquiring 50 game steps, the robot had learned the game rules perfectly, as shown in Fig. 9.

2) GAME 2: Muk-Chi-Ba game

The rules of Muk-Chi-Ba are explained in [19]. This game starts with a usual game of Rock-Paper-Scissors. The person who wins the Rock-Paper-Scissors plays the offense for the first round. The offensive player either changes or maintains his/her hand while simultaneously saying the name of the hand position. Here, the opponent also changes or maintains his/her hand at the same time.

If the opponent's hand is the same as that of the offensive player, the offensive player wins. If the two players' hands are not the same, the offense/defense is redetermined from the hand positions of both players based on the rules of Rock-Paper-Scissors and the next round begins.

Five minutes observation of two human participants playing Muk-Chi-Ba resulted in 111 game steps. As illustrated in Fig. 10, the robot learned the game rules completely after observing more than 70 game steps.

After observing two human participants play Rock-Paper-Scissors and Muk-Chi-Ba, the robot played the game against human participants. In each game, the robot played with one of the human participants for three minutes. The robot played the game flawlessly according to the learned game rules. When the human players violated a rule, the robot gave a warning by saying "You violated a rule! I win! "

3) GAME 3: Three-player Rock-Paper-Scissors game

The proposed approach is also applicable to games with more than two players. The rules for three-player Rock-Paper-Scissors are the same as those of the two-player game, with the game continuing until a single final winner is determined. We tested our algorithm on this game in a virtual environment. The game process was randomly created following the actual game rules. A total of 72 game steps (about 3 minutes) were generated and 3×72 steps were used for game learning. The results of these game simulations demonstrate the validity of our algorithms (Fig. 11).

4) GAME 4: Blackjack

To show the scalability of the proposed algorithm, we additionally tested on a different type of game. We selected the Blackjack, one of the most popular card games in the world. Unlike above games, the game rules of Blackjack are composed of visual information (number on the card) as well as sound and motion. In addition, the number of action combination (*nAC*) is far greater than that of above games because of the wide range of states of vision nodes.

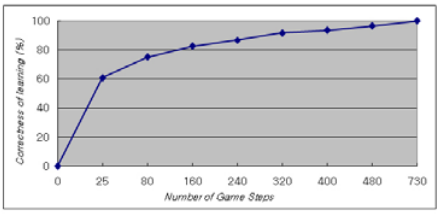
Before learning the game, we assume that the robot knows below basic information about this game.

- Sum of numbers on the cards is the key of the game.
- Each value of J, Q and K card is 10.
- Value of A can be 1 or 11.
- Not considering complex rules such as 'Double down' and 'Split.'

The game process was randomly generated following the actual game rules [20]. A total of 730 game steps (about 45 minutes) were generated and used for game learning. The results of these game simulations are described in Fig. 12. After watching 320 game steps (about 20 minutes), the robot learned more than 90% of the game. The robot can act properly even when it encounters unforeseen situation by an intelligent inference algorithm in the BNs. Details of the algorithm will be described in the future publication.

Considering that the velocity of learning is very fast at the

Pattern	Player 1 Sound	Player 1 Vision	Player 1 Motion	Player 2 Sound	Player 2 Vision	Player 2 Motion	Action Combination
P1	X	#	R	X	#	R	190
P2	# (B, H, S)	#	X	X	#	X	240
P3	X	#	R	X	#	X	240
P4	X	#	X	H	#	X	135
P5	X	#	X	X	#	# (R, F)	387
P6	W	#	X	X	#	X	1
P7	X	#	X	# (W, D)	#	X	1
	[Sound] B: Blackjack! H : Hit, S : Stand W: I Win, D: Draw		[Vision] # : value between 2~26 depending on the types of pattern		[Motion] R : Card Receive F : Card Flip		nAC = 1194



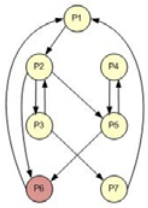


Fig. 12. Simulation results of the Blackjack game

early stage of observation (the robot learned more than 80% of this complex game in 10 minutes) and it becomes slower with the increase of number of observed game steps (observation time), we can expect that proper instructions together with observations might cut down the total learning time of complex games.

V. Conclusion

In the present work, we proposed a new game learning algorithm based on Bayesian networks and game pattern graphs. After observing human demonstrators repeatedly playing the game several times, the robot could learn the game interaction rules. A three-stage BN and a game pattern graph enable the robot's self-learning and playing. We tested the proposed algorithm on several famous games. The experimental results showed that the correctness of game learning increased fast in proportion to the observation time, with the robot learning the game rules completely within a few minutes of observation for the simple games. We also found that the robot can learn simple games without any information or human instructions, and learn more complex one with minimal information about the game. Having learned the game rules, the robot then played games with a human player robustly as humans do. From the results of experiments, we verified the effectiveness of the suggested algorithm, and hope this algorithm and the possible following works can significantly improve game interactions between humans and robots.

VI. Future Work

There are several issues left for future work. The robot must be able to learn the motion primitives by observing human gestures. In addition, learning the meaning of each behavior is very important to fully understand the social games and increase the interactivity with humans. We are now studying strategies by which the robot can adaptively react to different game players so as to increase the winning rate. We are also considering about expanding the suggested algorithm to multiple robot learning applications. Finally, using a combination of learning by demonstration and learning by instruction, future studies should consider more complex social games than the ones used in the present experiments. We will tackle these problems to realize more natural human–robot game interactions.

References

[1] K. Tanie, "Human Friendly Robotics – A Challenge to New Robot Applications," in *Proc. 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS99),* 1999, pp. 609

[2] D. Urting and Y. Berbers, "MarineBlue: A low-cost chess robot*" in Proc. IASTED International Conference on Robotics and Applications*, 2003, pp. 76-81

[3] A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, "RoboCup2005: Robot Soccer World Cup IX," 2006, vol. 4020

[4] D. C. Bentivegna and C. G. Atkeson, "Learning from Observation using Primitives," *in Proc. 2001 IEEE International Conference on Robotics and Automation (ICRA'01),* 2001, pp. 1988-1993

[5] D. C. Bentivegna and C. G. Atkeson, Gordon Cheng, "Learning tasks from observation and practice," *Robotics and Autonomous Systems,* vol.47, 2004, pp. 163-169

[6] C. Breazeal, A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, and D. Chilongo, "Tutelage and Collaboration for Humaoid Robots," *International Journal of Humanoid Robots,* vol.1(2), pp. 315-348.

[7] A. G. Brooks, J. Gray, G. Hoffman, "Robot's Play: Interactive Games with Sociable Machines," in Proc. 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology (ACE2004), 2004, pp. 74-83

[8] M. Ogino, H. Toichi, Y. Yoshikawa, and M. Asada, "Interaction rule learning with a human partner based on an imitation faculty with simple visuo-motor mapping," *Robotics and Autonomous Systems,* vol.54, 2006, pp. 414-418

[9] Aleksander Sadikov and Ivan Bratko, "Learning long-term chess strategies from databases," *Machine Learning,* 63(3), 2006, pp. 329-340

[10] Jonathan Baxter, Andrew Tridgell, and Lex Weaver, "Learning to play chess using temporal differences," *Machine Learning,*40(3), 2000, pp. 243-263

[11] Darse Billings, Lourdes Pena, Jonathan Schaeffer, and Duane Szafron, "The challenge of poker," *Artificial Intelligence,* 134(1-2), 2002, pp. 201-240

[12] Magee, D R; Needham, C J; Santos, P; Cohn, A G; Hogg, D C. "Autonomous learning for a cognitive agent using continuous models and inductive logic programming from audio-visual input," *in Proc. AAAI-04 Workshop on Anchoring Symbols to Sensor Data*, 2004, pp. 17-24

[13] K. B. Korb and A. E. Nicholson, "Bayesian Artificial Intelligence, " chapter 2, Chapman & Hall/CRC, 2004, pp. 29-47

[14] T. H. Cormen, C. E. Leiserson, Ronald L. Rivest, Clifford Stein, "Introduction to Algorithm," Section 24.3, *The MIT Press*, 2001, pp. 595-601

[15] http://www.robotis.com

[16] http://www.voiceware.co.kr

[17] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture models," IEEE Trans. On Speech and Audio Processing, 1995, Vol.3(1), pp. 72-83

[18] Timo Pylvanninen, "Accelerometer based gesture recognition using continuous HMMs," *Pattern Recognition and Image Analysis*, 2005, Vol. 3522, pp.639-646

[19] Wikipedia, "Muk-Chi-Ba," http://en.wikipedia.org/wiki/Muk-Chi-Ba

[20] Wikipedia, "Blackjack," http://en.wikipedia.org/wiki/Blackjack