KIKS Extended Team Description for RoboCup 2020

Yusei Naito, Shin Ohno, Yuta Imaeda, Akihito Odanaka, Yasutaka Tsuruta, Ryoma Mitsuoka, Taisuke Tane, Masato Watanabe, and Toko Sugiura[1]

National Institute of Technology, Toyota College, 2-1 Eisei-cho, Toyota, Aichi 471-8525, Japan
sugi@toyota-ct.ac.jp,
URL: www.ee.toyota-ct.ac.jp/~sugi/RoboCup.html

Abstract. This paper is used to qualify as participation to the RoboCup 2020 soccer small size league. Our team's robots and systems are designed under the RoboCup 2019 rules. In this paper for hardware, it is described about the various idea of dribble mechanism, whose importance was again emphasized in the final of RoboCup 2019 Sydney. In software, the implementation of UCT (Upper Confidence bound applied to Trees) in MCTS (Monte Carlo Tree Search) method was performed to achieve the action-decision of the motion for the robots. Moreover, we tried to use Deep Learning Library (Sony products) and Neural Network using GUI (Graphical User Interface) for layered design, learning and evaluation in AI system.

Keywords: RoboCup, small size league, autonomous robot, global vision, engineering education

1 Introduction

KIKS has been continuously developing the higher-performance hardware and smart AI system. In hardware, we worked on improving ball control performance, and implemented and examined them . In software, we examined and improved path generation for the opponent robots. In addition, Neural Network Libraries and Neural Network Console were introduced, and action-decision algorithm using neural networks and Monte Carlo tree search for robots were inplemented.

2 Mechanical design

In RoboCup SSL, to realize a strategy based on AI, it is necessary to have a mechanism to accurately kick a ball and a mechanism to accurately receive a ball. Our present robots, however, did not have sufficient performance of a mechanism for holding a ball, and often bounce off the ball during the game. Also, when a ball was kicked, the ball sometimes rolled in a different direction from the AI command because the ball was not held at the center of dribbling bar. If we can improve the ball holding ability, the dribbler will be an effective tactics like ZJUNlict in final game of RoboCup 2019 Sydney. In addition, when the dribbler's rotation can be sufficiently transmitted to the ball, a curve shot may be realized, which will be effective in expansion of AI strategies. Therefore, in this section, it focuses on dribbling performance which improves speed and stability for a ball, and describes an idea of new dribblers to improve the ball holding ability.

2.1 Idea of ball stabilization in present dribbler

In the RoboCup 2019 Sydney final, the importance for dribbling performance was once again demonstrated. In order to keep possession of a ball from opponent team and also to make a success for Auto Ball Placement, the dribbling mechanism that can continue to give a stable rotation of a ball is required. In our present dribble device, however, when a backspin is applied to the ball, even if the ball is stable at first, it gradually begins to vibrate and moves away from the roller. It is causally related to the fact that the ball bites too much into the dribbler and generates a large pressing force on the dribbler. As the result, the dribbler cannot withstand the mechanical load and it pushes the ball back. To solve this problem, for example, ZJUNlict team applied 3-point-touch system as shown in Fig.1(a)[1], and OP-AmP team removed the the chip-kick bar and attached a freely rotating roller as shown in Fig.1(b)[2].

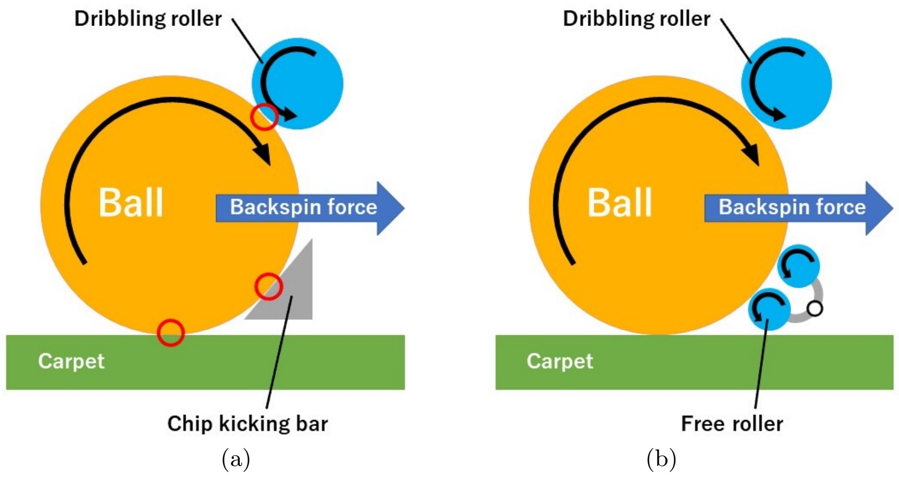


Fig. 1. Image of typical dribble structure for (a)ZJUNlict and (b)OP-AmP [2].

Our idea of dribbling device is shown in Fig.2. It has two dribbling rollers. By mounting the roller on a higher position, the ball is more strongly pressed

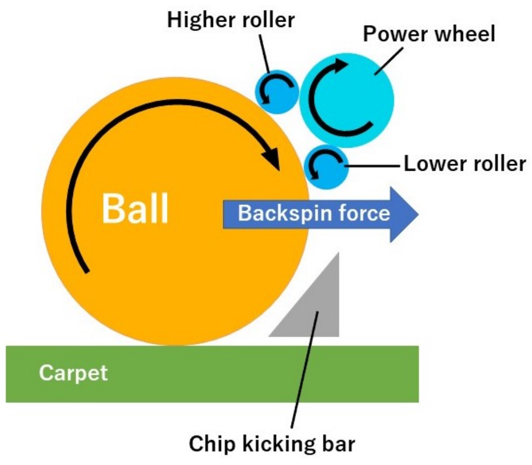


Fig. 2. Schematic drawing of idea for present dribbling device

against the carpet, and as the result, makes it difficult for the ball to leave the roller. Therefore, the roller can efficiently transmit rotation to the ball. But, as the contact point between the roller and the ball raises up, the forth with which the ball enters into the dribbling device increases. Thus, another roller is placed at the lower position that does not touch the straight-kicking bar. This roller stops the ball entering into the dribble device. The idea of this dribbling device will be able to stabilize a high-speed spin ball while keeping the chip kick function. In addition, a gap between the ball and the chip-kicking bar play as a role for no bouncing off the ball when catching it. The prototype dribbler used for the test is shown in Fig.3. The Material and diameter of roller used in prototype is tabulated in Table. 1.

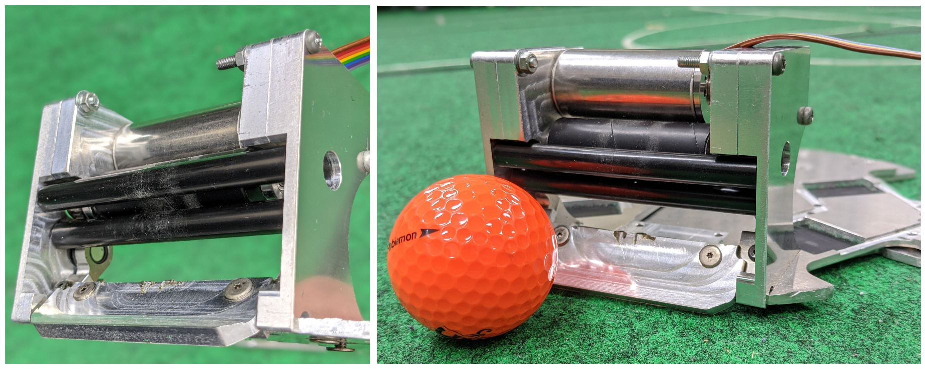


Fig. 3. Prototype dribbler used for the test.

Table 1. Material and diameter of roller using for the prototype dribbler

	Material	Diameter
Two small rollers	Urethane	8mm
Power wheel	Rubber	16mm

Two small rollers and a power wheel shown in Fig.2 transmit the force by friction contact with rubber. In fact, we tried to transmit rotation to the ball using a prototype. As the result, when the rotation speed exceeded around 2000[rpm], the ball and the roller slipped, and the rotation of power roller could not be transmitted sufficiently to a ball. This is probably because the contact surface between the ball and the rollers was small due to the small roller diameter, and the roller material was too hard. At present, there is no mechanism to adjust the angle of the dribble bar according to the position of the ball. Hereafter, we would like to make a confirmation of effectiveness of this idea by changing the roller diameter and material.

Furthermore, in order to realize the precise pass-play, we improved the accuracy for kicking by adding the spiral structure on dribbling roller as shown in Fig.4. This is similar to the idea of the ZJUNlict [1] and TIGREs Mannheim[3]. It allows to center the ball automatically and instantly. Because of this change, the accuracy for kicking as well as the probability of the Auto Ball Placement were improved.

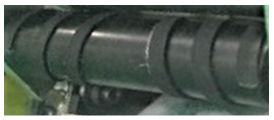


Fig. 4. Dribbler's surface with a spiral structure.

2.2 Idea of new dribbling device

Here is another approach to receiving a ball at the center of the dribble bar. The speed of the dribble roller is proportional to the radius. Therefore, by making the center thicker and the ends narrower, the velocity of the ball will add a force towards the center. In this subsection, we propose a mechanism that can change the dribbling-bar angle by using a parallel link mechanism and a servomotor. The 3D CAD image is shown in Fig.5. In this mechanism, when the ball is held, the position of the ball is recognized by infrared sensors, and the angle of the dribbling bar is adjusted based on the result. This makes it possible to keep the position of the ball at the center of the dribbling bar at any time.

In addition, even when the robot is dribbling, the angle of the dribbling-bar can be properly adjusted by using the parallel link mechanism. Therefore, it is considered possible to keep dribbling without releasing a ball. Since the effectiveness of those performances has not been confirmed yet, detailed verification will be carried out in the future.

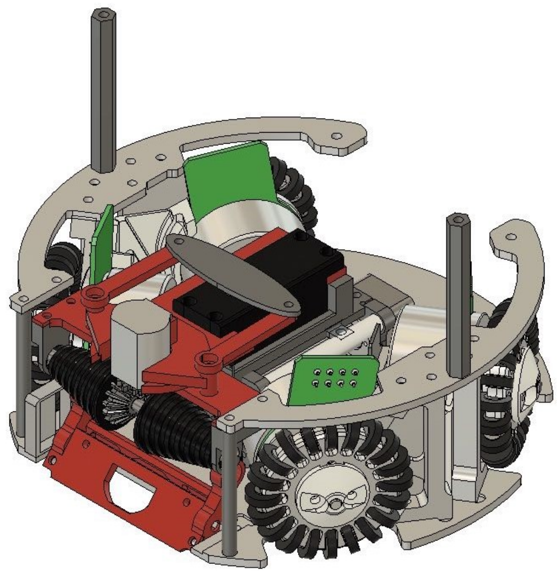


Fig. 5. 3D CAD concept image of new dribbler with parallel link mechanism.

2.3 Application to diagonal kick using movable dribbler

In RoboCup SSL, a wide choice for kicking mechanisms including straight kick, chip kick and curve kick is effective in expanding the range of AI strategies. The mechanism of the curve kick has been already reported by OP-AmP [4] and RoboTeam Twente [5], but we also considered its introduction. We propose a new method for curve kick using a movable dribbler. Using the prototype mechanism shown in Fig.6, we tried to realize a curve kick by applying a diagonal force to the rotating ball. It is possible to achieve by changing the angle of the dribbling-bar with a servomotor before kicking. It was confirmed that the trajectory of the ball after kicking can be controlled by adjusting the angle of the movable dribbler. But, it does not implement yet, because of some problems are occurred for mounting a dribbling motor and link mechanism compactly and for considering the shape and material as same as conventional dribbler. In the future, due to control the rotating speed of dribbler, the orbital radius of the ball will be able

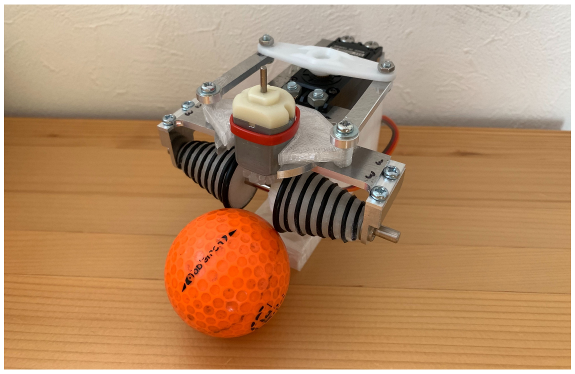


Fig. 6. Experimental prototype dribbler for the test.

to change and expand the range of tactics. It will also give advantage in penalty kick.

3 Software design

We examined and improved the path generation and ball competition, which have been problems since last year. We aim to realize a smart game. In recent years, the field size and the number of robots are expanding in RoboCup SSL. The conventional decision-making algorithm has problems such as not considering future prospects and poor expandability. It is impossible to realize cooperative actions such as passes during in play, and resulting in a low score rate. Therefore, we tried to apply a technique that combines Monte Carlo tree search and neural network, which has been successful in the field of game AI in recent years. We tested this algorithm in practice games.

3.1 Improvement of path planner

In previous system, robots were moving by path generation, but their performance was poor. Therefore, there were many collisions with opponent robots. In this subsection, we describe about improvement for our route generation. The current route generation method only generates a path in one frame, and does not consider the moving direction and speed of the partner robot. Therefore, the moving position is predicted from the speed and the distance of the obstacle, and the path is generated based on the predicted position. First, a prediction time T is considered based on the distance to the obstacle closest to the target robot. As the distance to the obstacle approaches, the time to collision decreases. If T is fixed, the risk of collision increases. Next, assuming that the obstacle moves linearly at a constant speed, the position after T seconds is predicted. Finally, a path planner is examined using the predicted position as an obstacle. As the result, the motion corresponding to the action of the obstacle will be realized, as shown in Route B instead of Route A in Fig.7.

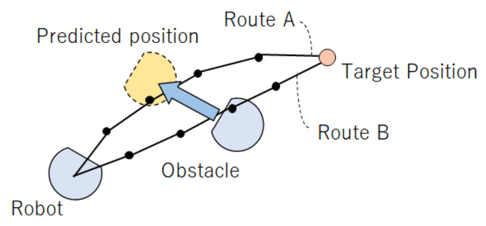


Fig. 7. Improvement of path planner.

3.2 Introduction of Neural Network Libraries / Neural Network Console

We have introduced the Neural Network Libraries (NNabla) [6] and Neural Network Console[7] provided by Sony Network Communications Inc. to implement Deep Learning. The neural network is designed, learned and evaluated using the Neural Network Console, and inference is performed using C++ API of NNabla. Neural Network Console can be operated with GUI, and trial and error is possible efficiently.

3.3 Decision-making algorithm using Monte Carlo Tree Search and Neural Network

When a robot plays soccer, there are various action options such as pass, shoot, and dribble. Significant rule changes in recent years have increased the number of robots and the size of the field. Rule changes may continue in the future. Therefore, decision-making for motion of the robots considering future prospects will be more important in strategy. We implemented an algorithm that combines Monte Carlo Tree Search (MCTS) and Neural Networks with reference to AlphaZero[8, 9] developed by DeepMind. MCTS is a search method based on simulation results. This method has been used in the field of Go etc.[8, 9], but we applied it to soccer by inferring the success probability of action.

Implementation of MCTS The nodes of the game tree have information of the position, elapsed time, Action, Reward, and Visit-count for the robot in the field. The node transitions by Action. At this time, the transition probability to each node is derived by inferring the success probability of Action using a Neural Network. This inference is also applied to the MCTS Rollout (random simulation), and it is possible to obtain the expected value of the reward and to search for promising actions. Typical game tree is shown in Fig.8.

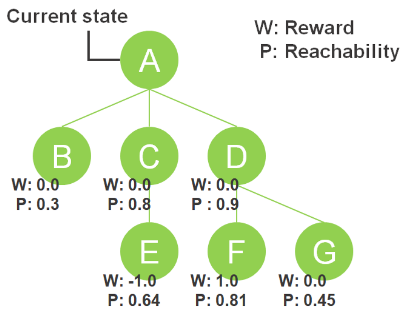


Fig. 8. Typical game tree for action-decision.

MCTS is an algorithm that expands and searches a tree based on random sampling. Compared to the $\alpha\beta$ method generally used in Japanese chess (Shogi), it has the advantage that there is no need to create and adjust complex evaluation functions. MCTS consists of five elements, i.e., *Selection*, *Expansion*, *Evaluation*, *Backup*, and *Play* for the process as shown in Table2. First, the algorithm creates a root node with the state of the observed field. Next, selection, expansion, evaluation, and propagation from the root node are repeated according to the search situation until a certain time has elapsed. After that, the node with the highest number of selections is selected from the child nodes connected to the root node, and the action to execute is made decision.

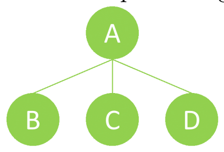
Table 2. Process elements on the Monte Carlo Tree Search

Process	Definition
Selection	Select the child node with the highest UCB1 value.
Expansion	When a ball is in playing field even if the search reaches a leaf node, the node is extended in case that the Visit-count is greater than a certain value.
Evaluation	When a ball is in playing field even if the search reaches a leaf node, the result of Rollout (random simulation) from that state is added to the reward of the node in case that the Visit-count is less than a certain value.
Backup	When a ball is not in playing field and the search reaches a leaf node, the state evaluation value is propagated to the parent node.
Play	After the search is completed, the node with the highest Visit-count is selected, and the action to be performed is determined.

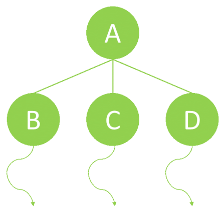
The UCB1 (Upper Confidence Bound) [10] described in eq.(1) was used as the selection criteria for child nodes to be searched. UCB1 is well known for the multi-armed bandit problem, and it is possible to search in consideration of the evaluation value and accuracy for decision branches. where, w_i and n_i show the Reward and the Visit-count for nodes i, respectively. The t indicates the Visit-count for all of child nodes, c is the constant. In eq. (1), the first term indicates the average of the reward, and the second term means the value that gives the reliability of the evaluation. The second term becomes larger as the Visit-count (evaluation value) to the node is smaller than the total Visit-count, and is easily selected. That is, the evaluation value increases as the reliability of the first term decreases.

$$UCB_i = \frac{w_i}{n_i} + c\sqrt{\frac{2log(t)}{n_i}} \tag{1}$$

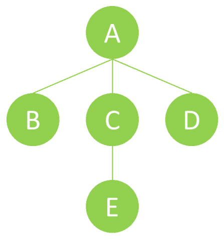
(a)Creation of a root node representing the current state



(b)Extension from root node to node



(c)Selection of the largest value of UCB and evaluation by Rollout



(d)Extention from a node whose evaluation number reaches a certain level to a node

Fig. 9. MCTS execution procedure

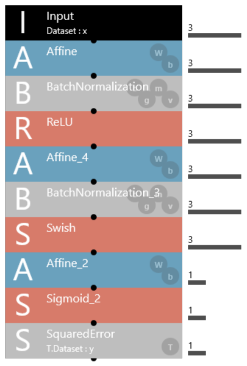
The ϵ-greedy method, in which the evaluation value is randomly selected with a certain probability, was also studied. In this study, however, UCB1 that mathematically guarantees convergence to the optimal solution was used. The execution procedure of MCTS is shown in Fig.9. The update of the evaluation value and the expansion of the tree are repeated in Fig.9 (c) and (d). Finally, the action that transitions to the child node with the highest Visit-Count is executed except the root node. For example, in Fig.9, an action to transition to any of nodes B, C and D is executed.

Neural network design The success probability of actions such as pass and shoot is inferred by a neural network. The processing for the enemy robots is performed one by one, and the inferred values of all the robots existing on the field are combined. Also, when creating teaching data and performing inference, it was assumed that there was no enemy robot farther than the target position at the start position of the action. That is, the inference was based only on the distance to the robot located in front of the target position. For the input data, it was used the ball speed, the distance between the ball and any enemy robot, and the angle between the ball and any enemy robot. The output is that the ally robot can either act or not, without intercepting the ball by the enemy robots. The reachability inference NN described here was composed of all connected layers, and several NNs with different numbers of layers, nodes, and activation functions were compared.

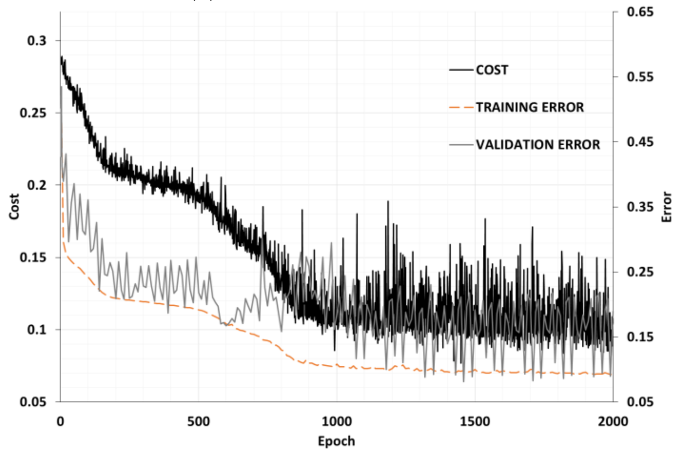
The success probability of Action is inferred by a Neural Network structure with all layers of connections as shown in Fig.10(a). This network infers the probability that an enemy robot will get a moving ball. This inference is applied to all enemy robots on the field to calculate the probability that the ball can move at any position and any speed.

Learning for Neural Networks More than one hundred learning data were obtained from the game simulation on grSim[11], and sent to Neural Network Console. Execution example of learning process is shown in Fig.10(b). For the learning, it was performed with a batch size of 32 and 2000 epochs.

Performance evaluation of neural network Figure10 shows the structure and learning curve of the introduced reachability inference NN. In Fig.10 (b), the horizontal axis shows the number of epochs (generations), the left vertical axis shows the output of the cost function, and the right vertical axis shows the error rate. TRAINING ERROR and VALIDATION ERROR indicate error rates for learning data and for verification data, respectively. The introduced NN has two hidden layers. The first hidden layer has three nodes, and the activation function is ReLU. The second hidden layer has three nodes, and the activation function is Swish. The time until completion for learning was 36.2sec. Obtained probability are tabulated in Table.3. The accuracy of the inferred value was about 84%, which was enough for use in the actual games.



(a) Neural Network structure



(b) Typical Learning Curve

Fig. 10. Neural Network structure (a) and learning process (b).

Table 3. Probability predicted by Neural Network on Confusion Matrix (50 samples)

Data \ Predicted probability	<0.5	≥0.5	Recall
FALSE	<u>18</u>	7	0.72
TRUE	1	<u>24</u>	0.96
Precision	0.9473	0.7741	
F-Measures	0.8181	0.857	
Accuracy	0.84		
Average of Precision	0.8607		
Average of Recall	0.84		
Average of F-Measures	0.8376		

On the other hand, Table4 shows the accuracy of NNs with different numbers of node in the hidden layers. As the result, it can be confirmed that the accuracy is lower than that of introduced NN.

Table 4. Comparison of Accuracy for NN under the various conditions

Condition	First hidden layer: 8, Activ. func.: ReLU Second hidden layer: 4, Activ. func.: Swish	First hidden layer: 5, Activ. func.: ReLU Second hidden layer: 3, Activ. func.: Swish	First hidden layer: 3, Activ. func.: ReLU Second hidden layer: 3, Activ. func.: ReLU
Accuracy	0.76	0.76	0.64

Performance evaluation of decision making algorithms An example for the action of this algorithm on grSim is shown in Fig.11 (Observation with AutoReferee [12]). The circle in the lower center represents the ball, and the markers numbered B or Y indicate the robots of each team. First, at the lower left in Fig.11, it can be seen that the B5 robot is moving toward the opponent's goal after passing the ball toward B1. After that, by receiving the ball again from B1 in front of the goal (ducking and weaving the opponent robots, Y1 and Y4), it is found that the B5 robot will attack toward the goal. We introduced this algorithm to real game (RoboCup JapanOpen 2019) and confirmed that an effective pass play could be realized. Furthermore, we plan to use the selection probability and evaluation value for Action as the inference value of Neural Network. As the result, the efficiency of the tree search is improved, and the range of the decision branch for the action will be wider.

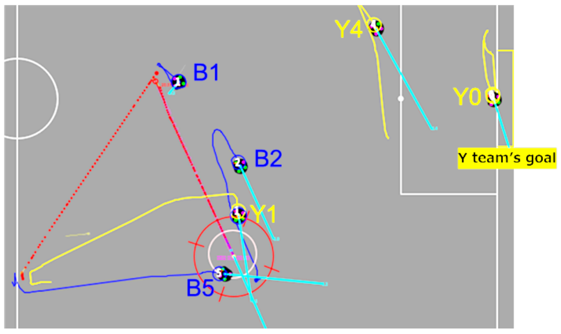


Fig. 11. An example for the action of this algorithm on grSim.

Planned Neural Network structure is shown in Fig.12. In this network, we consider a matrix of 1m intervals. It receives three channels as input, i.e., the positions for a ball and both team's robots, and outputs the expected score and the distribution of the selection probabilities for action candidates. One of the problems of this method is that the computational complexity is greater than conventional methods. As the solution, now we are studying about the introduction of GPU for Neural Network processing, and the separation and parallelization of this method.

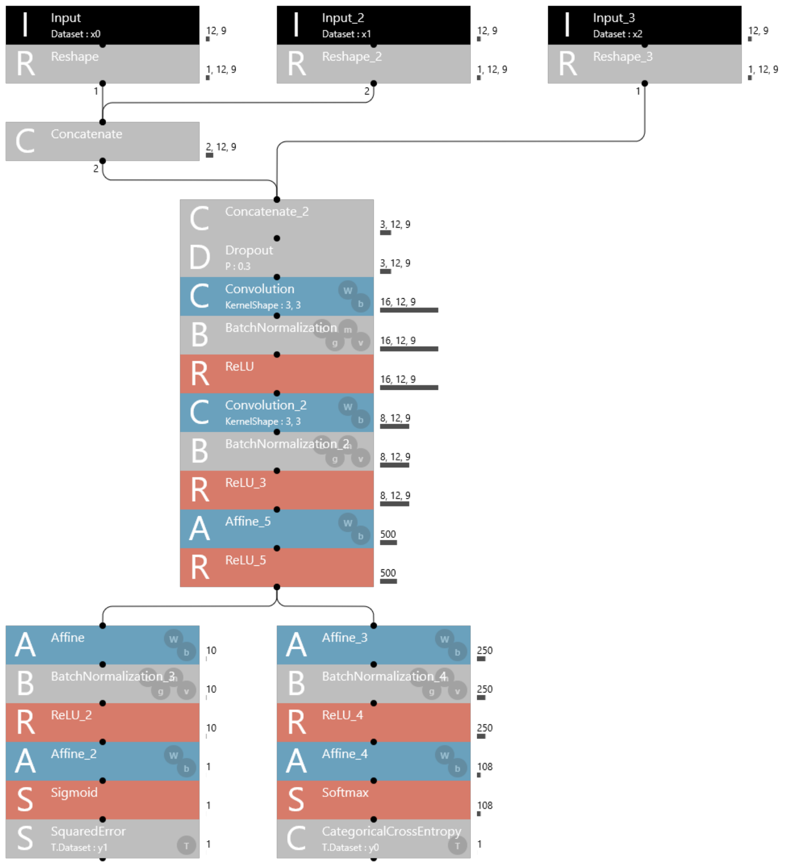


Fig. 12. Planned Neural Network structure for inferring expected score and the distribution of the selection probabilities for action candidates.

3.4 Game reporter for the audience

In SSL game, there are many complicated and specific rules for the league, and because of the offense and defense change quickly, it is difficult to understand the game situation for the audience. Therefore, we studied about the game reporter system that automatically explains the game situation of SSL to the audience by artificial voice. Rough process flow for the game reporter is shown in Fig.13.

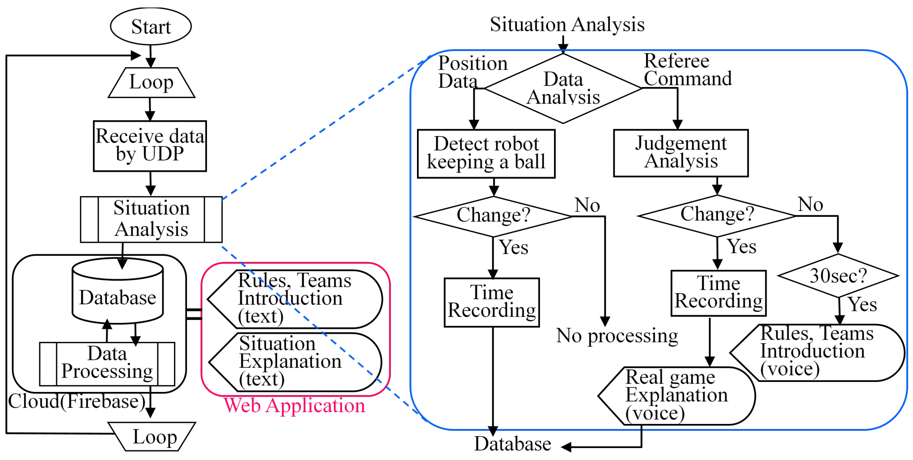
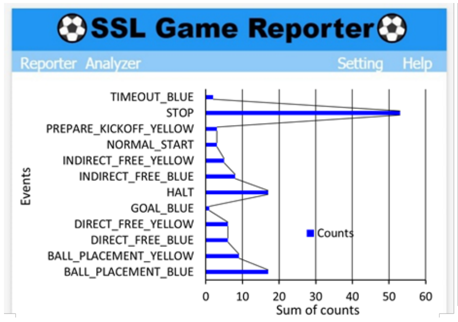
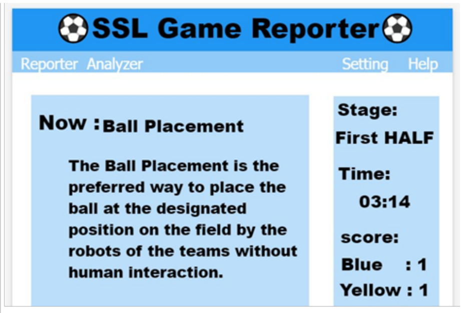


Fig. 13. Rough process flow of game reporter.



(a)



Now : Ball Placement

The Ball Placement is the preferred way to place the ball at the designated position on the field by the robots of the teams without human interaction.

Stage:
First HALF

Time:
03:14

score:
Blue : 1
Yellow : 1

(b)

Fig. 14. Typical image on the game reporter, for counts of events in each team (a) and explanation to speech (b).

Parsian reports the game reporter function as one of the log analyzers in 2019 ETDP[13]. We have created a smartphone application that provides game event information to the audience. To explain the game situation, it is necessary to create a rule base corresponding to the situation. It is especially important to understand the features of the SSL game and the differences from human soccer.

We investigated the number of events such as goals, passes and rule violations from the game log. For example, the number of events in each team at the SSL Final at RoboCup 2019 is shown in Fig.14(a). This result will be helpful in explaining the difference from the actual human soccer game. Figure14(b) shows an application screen image for visually displaying the game situation. In the future, we will introduce a function to explain the reason of the foul, the situation of the game and competing teams introduction by local language, and create a rule base for giving an appropriate explanation according to the progress of the game. We think that the explanations in local language are particularly effective in attracting children's interest.

References

1. Zheyuan Huang, Lingyun Chen, Jiacheng Li, Yunkai Wang, Zexi Chen, Licheng Wen, Jianyang Gu, Peng Hu, and Rong Xiong: ZJUNlict Extended Team Description Paper for RoboCup 2019, https://ssl.robocup.org/wp-content/uploads/2019/03/2019_ETDP_ZJUNlict.pdf (2019).
2. Takamichi Yoshimoto, Takato Horii, Shoma Mizutani, Yasuyuki Iwauchi, and Shota Zenji: OP-AmP 2019 Extended Team Discription Paper, https://ssl.robocup.org/wp-content/uploads/2019/03/2019_ETDP_OP-AmP.pdf (2019).
3. Nicolai Ommer Andre Ryll and Mark Geiger: TIGERs Mannheim Extended Team Description for RoboCup 2019, https://ssl.robocup.org/wp-content/uploads/2019/03/2019_ETDP_TIGERs_Mannheim.pdf
4. Takamichi Yoshimoto, Takato Horii, Shoma Mizutani, Yasuyuki Iwauchi,Yutaka Yamada, Kousei Baba, and Shota Zenji: OP-AmP 2017 Team Discription Paper, https://www.robocup2017.org/file/symposium/soccer_sml_size/Robocupssl2017-final9.pdf (2017).
5. Cas Doornkamp, Zahra van Egdom, Gaël Humblot-Renaux, Leon Klute, Anouk Leunissen, Nahuel Manterola, Sebastian Schipper, Luka Sculac, Emiel Steerneman, Stefan Tersteeg, Christophe Vanderwalt, Wouter van Veelen, Haichuan Wang, Jeroen Weener, Jelle Zult: RoboTeam Twente 2018 Team Description Paper,https://ssl.robocup.org/wp-content/uploads/2019/01/2018_TDP_RoboTeam_Twente.pdf (2018).
6. Neural Network Libraries, https://nnabla.org
7. Neural Network Console, https://dl.sony.com
8. David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, Demis Hassabis: Mastering the game of Go without Human Knowledge, nature, 2017. https://deepmind.com/research/publications/mastering-game-go-without-human-knowledge
9. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu,Thore Graepel & Demis Hassabis: Mastering the game of Go with deep neural networks and tree search, nature, 2016. https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf

10. Peter Auer, Nicolò Cesa-Bianchi, Paul Fischer: Finite-time Analysis of the Multi-armed Bandit Problem, https://link.springer.com/article/10.1023/a:1013689704352
11. Monajjemi, Valiallah (Mani), Ali Koochakzadeh, and Saeed Shiry Ghidary; grSim – RoboCup Small Size Robot Soccer Simulator, In Robot Soccer World Cup, pp. 450-460. Springer Berlin Heidelberg, 2011.
12. TIGERs Mannheim AutoReferee, https://github.com/TIGERs-Mannheim/AutoReferee
13. Kian Behzad, Elham Daneshmand, Nadia Moradi, Mohammad Reza Kolani, Mahdi Hajimohammadi Onidin, Yasamin Alizadeh Gharib, Atiyeh Pirmoradi, Mohammad Mahdi Rahimi, Mohammad Mahdi Shirazi, and Mohammad Azam; PARSIAN 2019 Extended Team Description Paper https://ssl.robocup.org/wp-content/uploads/2019/03/2019_ETDP_Parsian.pdf (2019).