PARSIAN 2017
Extended Team Description Paper

Mohammad Mahdi Rahimi, Mohammad Mahdi Shirazi, Maziar Arfaee, Mohammad Amin Najaf Gholian, Amir Hossein Zamani, Hamed Hosseini, Fateme Hashemi Chaleshtori, Nadia Moradi, Atousa Ahsani, Mahmoud Jafari, Amin Zahedi, Parsa Abdollahi, Alireza Zolanvari and Mohammad Azam Khosravi

Electrical Engineering Department
Amirkabir Univ. Of Technology (Tehran Polytechnic)
424 Hafez Ave. Tehran, Iran
{mmrahimi,mhmmdshirazi,maziar9033063,ahz.7597,hellihdhs,
hashemi96,nadiamoradi,ahsani,zahediamin,p.abdollahi,
alirezazolanvari,m.a.khosravi}@aut.ac.ir

Abstract. In this paper detailed description of Parsian robots' hardware improvement, as well as the software architecture with focus on new improvements that have been made since last year, is represented. Improvements and developments that seemed innovative and useful in hardware like *fault detection*, *two way communication* and *dribbler system* and also improvements in software such as *pass and interception* skills' behavior, *adaptive* attack strategy for regular game and *reactive* offense and defense for free kicks will be discussed in detail.

Keywords: graph theory, automata theory, machine learning, machine vision

1 Introduction

The Parsian small size team, founded in 2005, is organized by Electrical Engineering Department of Amirkabir University of Technology. The purpose of this team is to design and build small size soccer robots compatible with International RoboCup competition rules as a student based project. We have been qualified for eleven consequent years for RoboCup SSL. We participated in years 2008 to 2016 RoboCup competitions. Our most notable achievements was Parsian's first place in RoboCup 2012 SSL's Passing and Shooting and RoboCup 2013 SSL's Navigation challenge. In this paper we first represent some mechanical changes of our robots in Section 2.1 and then features developed for electrical design in Section 2.2 and in Section 3 we discussed about software improvements in control and behavioral level and at last introduce the *log analyzer* made in this year in Section 3.4.

2 Hardware

2.1 Mechanical Design

Last year, our mechanical team started to design a new version of robots (Fig. 4) which has many benefits compared to the previous version (Fig. 1). In this version, about 90% of parts had been changed mechanically and the details is represented . The kicking system including *flat kick and chip kick* remains unchanged. We use a cylindrical solenoid for flat kicking which can kick the ball with the velocity up to 14 m/s and the chip kicking system is a flat solenoid.

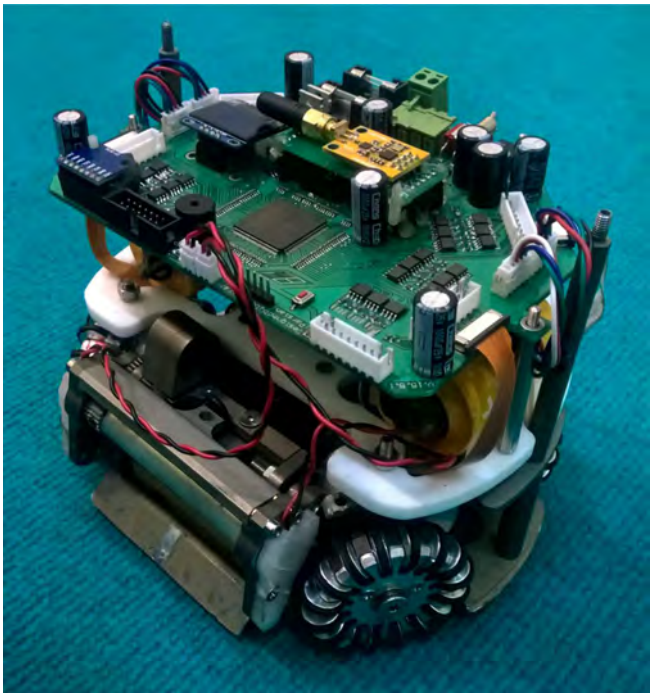

Fig. 1. The previous version of Parsian robots

Power Transmission System: Propulsion power of the robots is supplied by EC45 MAXON BLDC motors with an idle speed of 6720 rpm and nominal torque of 97.1 mN.m and power of 50 W instead of the 30W MAXON motors used in the previous versions. The power is transmitted to the wheels by a 3.6:1 gear transmission. In the gearbox, we use a spur gear attached to the motor shaft and an internal gear for wheels. The structure of wheels is same as the previous one.

Dribbler System: The previous version of our dribbler system had good performance but because of the complexity of mechanical structure, it was very difficult and time consuming to assemble the whole system and if there was a problem, it took too much time to be fixed (Fig. 2). Therefore the main purpose of the new design is simplicity along with good performance.

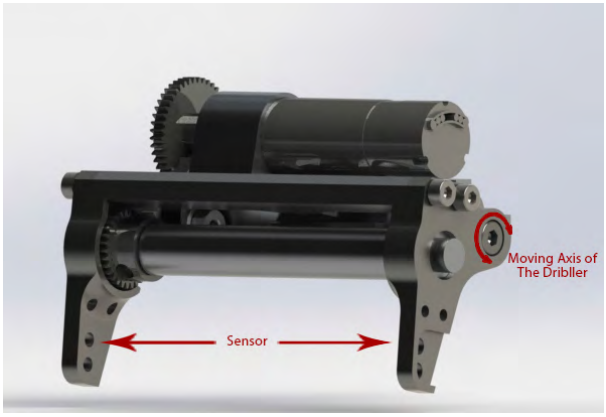

Fig. 2. Previous version of dribbling system

Dribbler system has three major duties that are (a) *ball detection* (b) *ball damping* (c) *ball spinning*. In our robots, *ball detection* is handled by *IR sensors* which is fixed in dribbler bases. *Ball damping* and good *ball spinning* depends on each other. During *ball spinning*, the spinner pulls the ball into the dribbler with a high rotational speed which cause a small displacement of dribbler arms and ball towards the center of the robot that should be considered in dribbler designing. In order to have an excellent spin, the ball has to be in contact with ground, spinner and the chipper head during the spinning. So the point is optimizing the dimensions with these two considerations. In the other hand, the best location for spinning the ball into the dribbler is the top of the ball but due to the existing rules, we should find the allowable contact point which is the nearest location to top of the ball. Our new dribbler arm has one DOF which is a rotational movement and therefore for the *ball damping*, we use a smooth torsion spring. This spring also enhance the spinning performance because of vibration absorption.

In this version of dribbler, T-motor MN1806 is used as spinning motor with a maximum speed of 36000 rpm (which is in control with input voltage) instead of MAXON EC16 motors. These new motors have some advantages and disadvantages in comparison to MAXON EC16 motors.

Advantages are (a) lower weight, (b) smaller size, (c) no need to gearbox, (d) very

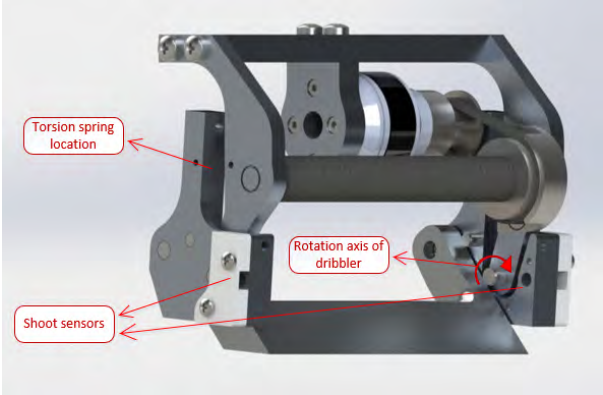

Fig. 3. New designed dribbler system

low cost, (e) very low vibration and high stability in high speeds and (f) lower sound and the disadvantage is lower torque.

The torque is transmitted to the spinning bar using pulleys and belt which has less vibration. The transmission ratio is 16:28. In order to compensate for lower transmitted torque, the rubber cover of spinning bar diameter was decreased from 16mm to 10mm (Fig. 3).

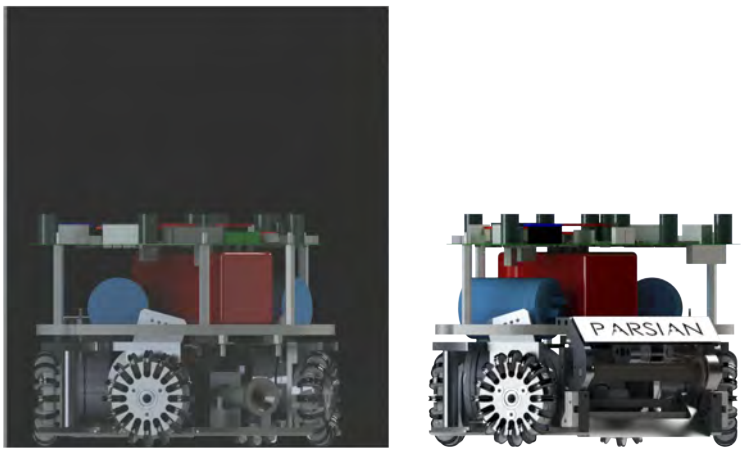

Fig. 4. New concept of Parsian robots

2.2 Electrical Design

Our main board was designed for Robocup 2016 and that was explained by details in Parsian 2016 ETDP [1] (Fig. 5). This year Parsian developed some features such as *two way communication* and *fault detection engine* that is described in detail in this paper.

Fig. 5. Parsian PCB boards

Two Way Communication: This year Parsian developed a new application for manage robots *health details* and we need some important data for manage robots behavior, therefore Parsian's *two way communication* is upgraded. The Packet sent from robot to *server* includes battery level, kick sensor, spin motor's current, robot ID, compass angle, *health data* and motor velocities. Previous year two NRF24L01 had been used for two way communication on each robot but this year it's done using just one NRF and the robot switches NRF mode between send and receive in certain duration of time to avoid *data loss* and *data conflict* between the robots.

Fault Detection Engine: This year a *fault detection engine* is added to Parsian robots that detects faults like motor failure such as wiring or hall sensor problems, battery, kick system and other failures. In case of emergency, this engine stops robot, for instance when battery voltage is in critical level. Some data that has been sent to main application are used for *debug and maintenance* some parameters during a real game.

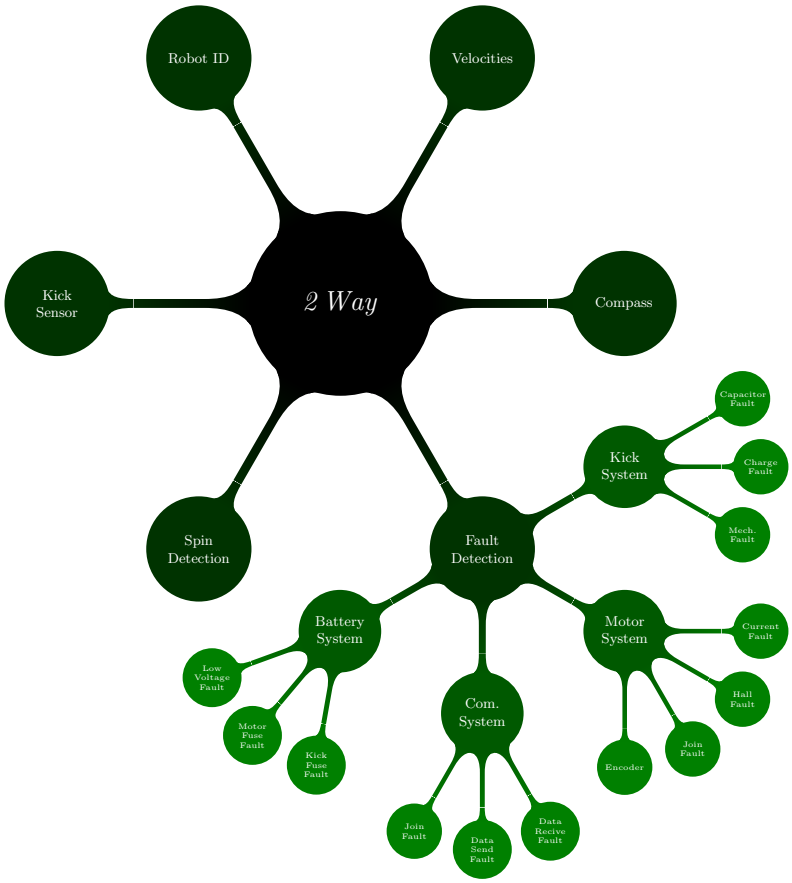

Fig. 6. Diagram of two way communication and fault detection

3 Software and A.I.

3.1 Control

New Algorithm for Moving Ball Interception: One of the most common problems in playing soccer is *ball prediction* and *interception*. This year Parsian designed a new engine for them. In this method ball is modeled and the engine can predict ball's position and velocity in the near future, also robot's abilities (maximum velocity, acceleration and joint speeds) is known and the engine can calculate required time for a robot to reach to a certain point. With these data, *the interception* searches ball's path with a fix step size (1 meter steps) to find the points that a robot can reach to them before the ball, when first point is found step size is reduced to 5 cm and a new search is began around the first suitable point and the best point (the most reliable point) is chosen (Fig. 7).

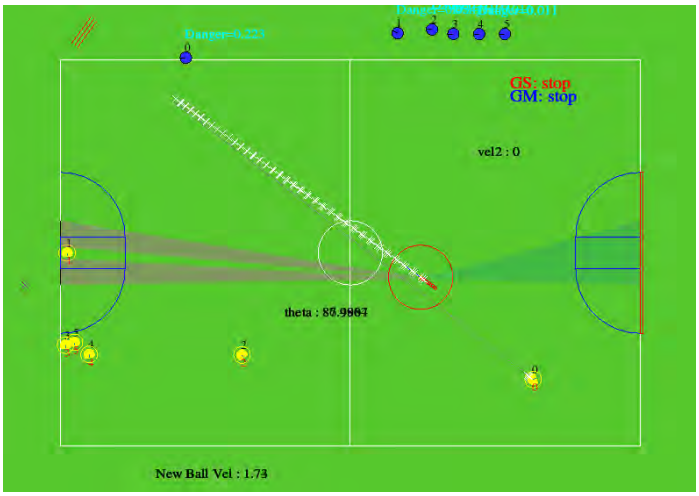

Fig. 7. Search ball path to find the best point to intercept ball

Spin Pass: During a small size soccer game it happens often that the ball is between two robots and they're trying to find a way to pass or dribble. This year Parsian produced a new algorithm to exploit this situation. In this situation the robot needs to snatch the ball from its opponent and make a unique situation to pass the ball to its teammate, so that the nearest opponent robot to the ball can't disturb its action. Therefore, the robot spins the ball with its maximum speed and tries to rotate around a certain origin that blocks the opponent robot to reach the ball, but the robot's rotation and its direction depends on the spin motor's current. In other words, the code analyses the spin motor's current and if the robot completely owns the ball, full rotation procedure is began; otherwise, the robot tries to rotate in small angles to catch the ball completely (Fig. 8, 9).

Fig. 8. Small rotation for snatch the ball

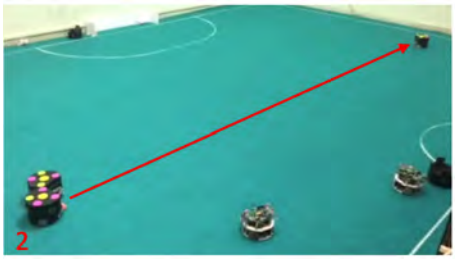

Fig. 9. Complete rotation for pass

3.2 Progresses in Offensive Plans

Improvements in Regular Game Behavior: The Parsian's offensive algorithm during regular game is a combination of previous years behavior and planner, that includes *static planner* used in 2015 [2] and *dynamic behavior* used in 2016 [1].

1. **Parsian 2015: Static Offense Planner**

According to Parsian's TDP [2] in 2015, we designed the regular game plans by the *visual-planner* and define a single behavior for any situation and *formation* in the field. The *static planner's* main benefit was always following steps that will score a goal and don't play with the ball causeless. The problems that Parsian's *static planner* faced is listed below:

- There's no coordination with the defensive team.
- There's no plan for the situation that we don't have ball possession (like counter attack).
- Sometimes direct shot behavior was more effective compared with passing the ball.
- There's no plan for contended situations.
- It needs too many plans to cover all the situations and behaviors.

2. **Parsian 2016: Dynamic Offense Behavior**

Parsian's *dynamic behavior* is designed in order to fix 2015's issues and mostly remove plan designing part by replacing an A.I. to generate plans. Here's a list of the solutions that dynamic behavior for *static planner* strategy:

- With dynamic assigning that we had in coach, any defensive agents needed to coordinate with offense added to offense team and plan the role given.
- *Dynamic behavior* make plans according to the game situation and *reactive* to opponent agent.
- There's no need to design plan anymore.
- This play was *configurable* to determine each behavior.

The problem that *dynamic behavior* faced:

- The goal of this behavior is to keep playing not scoring goal.
- The probability to have successful pass was low:
  1. Positions of *passer* and *receiver* agents was generated during the game and we can't tune pass skill for every situation.
  2. We don't fully profiled all robots' pass skill.
  3. This pass strategy should wait for the receiver agent to reach in receiving area and it wasn't fast enough.

So the result of this play with different configurations was two kind of attack:

- Mostly pass and fail at the end without scoring goal.
- Just keep forward attacking and positioning the agent stand in reflect positions.

And both behavior wasn't what we want from our dynamic attack, so we tried to use *human intelligence* in design and aiming plans to goal and use *reactive behavior* to execute this plans considering game's situations.

3. Parsian 2017: Adaptive Behavior with State Machine Designer

In This year, we focused on having a *state machine* that always end to score a goal. The idea firstly brought forth in 2013 [3]. There's two challenges, first we need to define states and design the state machine and second we should find the best edge to cross to reach the end points.

I Define States and Adaptive State Machine

States are defined as a condition that are clear and easy to detect and changing the state can be done by maximum one action for each of our agents. States should cover all situations of the game and not be dependent on opponent agents' position. Therefore we have full control on choosing states and changing them. An *adaptive state machine* is a type of state machine, made of some states and a relation between them, that can keep track of itself and use these data to make a better decision for the next time. A state is a pair of *formation* of our agents and ID of the ball owner agent. Note that this machine is used for attacking, so the ball is in our control and when we lose the ball, the program jumped out of the machine. The terminal states are situations that the ball is in opponent goal. When we get the ball, our formation is detected and get matched with one of our states. Then *state machine* change from one state to another so all the formations we get will be matched with one of the predefined states. States are designed by ssl-fedit and next, based on a configuration file, we build the state machine's edges.

II Giving Weights to Edges and Finding the Best Path

Edges is put between the current state and other neighbor states and weights is given to them to choose the best. Weight of an edge is sum of *reactive*, *configurable* and *adaptive* parameters:

- *Reactive* weight depends on opponent robots' positions and opportunity of kicking in that state.
- *Configuration* weight is given to an edge by developer. With this our attacking strategy can be set manually before the game.
- *Adaptive* weight is given to a plan based on experience, for instance when a plan fails repeatedly, its *adaptive* weight decreases.

Reactive Behavior for Free Kicks: According to Parsian's 2016 ETDP [1] we used *visual-planner* to design static and time based *free kicks* plans, for this year we've made some changes inside *visual-planner* and *free kick* behavior that make plans more flexible and *reactive* to opponents. In the following part, changes made in *visual-planner* is presented. After that we describe how we take advantages of them.

1. **Visual Planner Changes**

- Save and export plans with JSON format for easy human-readability and fast changing.
- Send plan book over network using Google Protocol Buffer.
- Add chance parameter to each plan.
- Design *open-end* plan (not determine receiver agent).

2. **Plan Execution**

- Reading plan over network, enable us to *hot-swapping* plans for editing.
- Execute plan by events instead of time for more *coordination* specially for pass and receive.
- If the plan is designed *open-end* then choosing the final receiver agent depends on opponent's marks and defense agents and it makes plans more unpredictable.

3.3 Defense Plans Improvements

The most basic aim of defending is preventing from giving opportunity to opponent agents to score a goal. In last year we only used the *Zone plan* for defending, but this plan hadn't enough efficiency, So we write the new algorithms *Block pass* and *Block shoot* tested in different competitions successfully, and also we have created a management system to choose one of these three plans and use them in suitable situations which result in having a defense system that varies according to the opponent offensive formation that can be changed in the middle of the game at once. This switching system is based on following parameters:

– Opponent's pass, shoot and movement skills.
– Areas which opponent team trying to attack from.
– Type of plans they use in play off.

Fig. 10 and 11 represent how *Block pass* and *Block shoot* works (we are the blue agents):

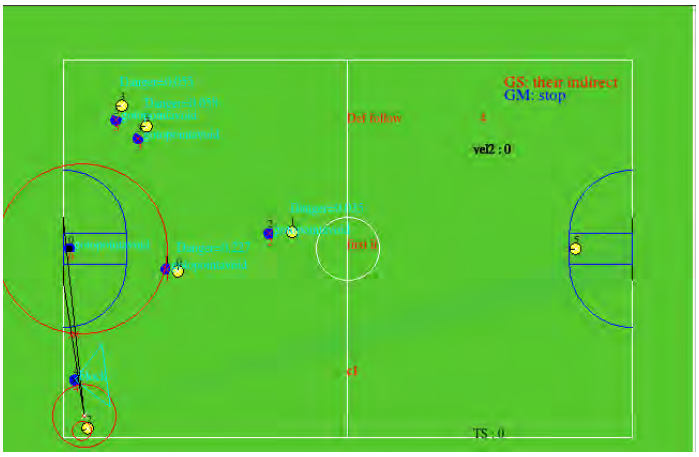

Fig. 10. Block shoot

Dynamic Assigning for Agents and Plans: In previous years we choose defense agents after each stop time, this make it difficult to cover goal during pass whereas we could not reach to definite point on time. The solution we offered is using *Dynamic robot role assignment* by considering the ball and our agents.

Attack Plans by Using the Defense and Goalkeeper: Since the Parsian Robotic Team is mostly focused on the defensive plans in proportion rather than offensive plans, we decide to make a big change in offense strategy which is explained in detail in Section 3.2 . Also we decided to start offensive plans from defense in order to throw the ball from our penalty area with the determined aim to make a offensive plan.

For instance when ball is in our penalty area and is caught by goalkeeper, goalkeeper's role changes to *playmaker* and stays in this role. With this role goalkeeper robot is inside *offensive team* having more coordination with offensive plan. After *playmaking* function is done, the role of *goalkeeper* changes back and return to *defensive team* [4].

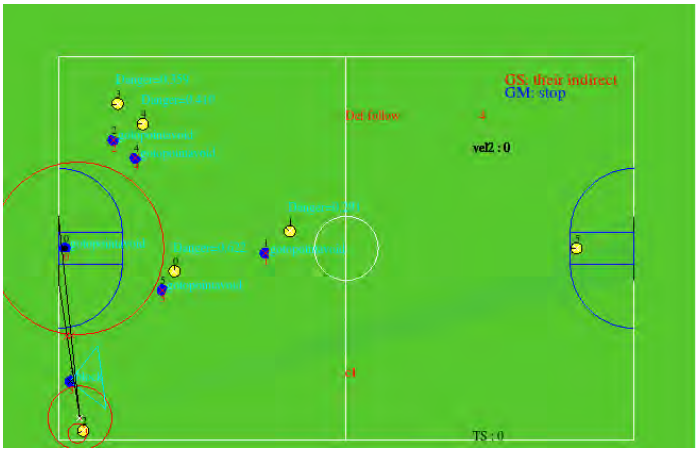

Fig. 11. Block pass

3.4 Log Analyzer

We use our specified *logger* to store match events and some other data that can help finding out useful information about the match. Our Logger encompassed vision data, referee commands and logging data. Vision data log are the filtered vision data we generate in our code. We also store referee commands therefore we can see them while replaying the logs. Debugs and draws are a part of our logging data to simplify debugging. The data we used to store were difficult to be analyzed so we created new log files for *robots info*, *control data*, *behavioural data* and *robot commands* to store useful data in a better format.

We analyze these log files to make report and turn them into useful knowledge to use them in (a) robot (b) control (c) behaviour (d) coach levels.

(a) By analyzing *robot info* and *robots commands* we collect and merge information about robots. For example how long our batteries can work or which commands cause harm.
(b) In control level we use *control data* and robot commands for profiling motion and kicks.
(c) In behavioral level we analyze our *plans* and weight them; for instance after each free kick we can make a report of our failed shoots and the places the robots has kicked the ball.
(d) In coach level we analyze general match information such as *ball possession*, the number of free kicks and percent of ball existence in each region of field.

4 Publication

4.1 ssl-visual-planner [1]

A user friendly software to arrange both dynamic and static plans for SSL. Most important improvements from this year is exporting plan in JSON format and send them over network.

4.2 ssl-fedit [2]

ssl-fedit is an *open-source* formation editor that presented in Parsian's ETDP in 2014 [5] and originally developed by Hidehisa Akiyama [6]. With Delaunay triangulation function expression model, developer can intuitively adjust the placement of the player on the GUI. We propose a modification to implement it in the offensive situation positioning of our small size soccer robots.

4.3 grSim [3]

The well known small size league simulation developed in Parsian [7]. This year we've made some changes and improvements such as four camera functionality with overlaps, compatibility with macOS, using a new ssl-vision protobuf and also fixing some physical parameters.

Our future work for grSim is to make it a *continuous integration framework* to test each feature while developing them and also add *mix-team* control on robots that enable more teams to get ready for this *technical challenge*.

5 Conclusion

This year our hardware's changes aim to stabilize robots with detect and monitoring their *faults*. In software part we improved some of our skills' behavior used mostly during the game such as *pass* and *interception*. In defense plans, we developed our last year defense by adding some new plans and a management system to handle them. In attack we have created a new system, based on our previous years experiences that needs more effort to be completed. Also we present *log analyzer* as a new feature to collect useful knowledge about our behaviors. Finally we represent our publication and future works that can be done, mostly focused on *grSim* to make it more flexible and use it for *continuous integration framework*.

[1] https://github.com/hamidrezakks/ssl-visual-planner
[2] https://github.com/mahi97/ssl-fedit
[3] https://github.com/mani-monaj/grSim

References

[1] M. M. Rahimi, M. M. Shirazi, P. Dajkhosh, A. Zolanvari, M. Arfaee, H. Kazemi Khoshkijari, A. Abbasi Fashami, A. Saeidi Shahrivar and M. A. Khosravi, "Parsian Extended Team Description for RoboCup", 2016.

[2] A. Zolanvari, M. M. Shirazi, S. P. Dajkhosh, A. M. Naderi , M. Arfaee, M. Behbooei, H. Kazemi Khoshkijari, E. Tazimi, M. M. Rahimi and A. Saeidi Shahrivar, "PARSIAN Team Description for RoboCup", 2015.

[3] S. M. Mohaimanian Pour, V. Mehrabi, A. Saeidi, E. Sheikhi, M. Kazemi, A. Pahlavani, M. Behbooei and P. Ghanbari, "PARSIAN extended team description paper for RoboCup", 2013.

[4] J. P. Mendoza, J. Biswas, D. Zhu, R. Wang, P. Cooksey, S. Klee and M. Veloso, "CMDragons 2015 Extended Team Description Paper", In RoboCup, 2016.

[5] A. Saeidi, M. H. Malmir, M. M. Shirazi, M. Behbooei, S. Boluki, M. Kazemi, S. M. Mohaimanian Pour, P. Ghanbari, S. Jamshidiha, P. Dajkhosh and A. Zahedi, "PARSIAN team description paper for RoboCup", 2014.

[6] H. Akiyama,I. Noda, "Multi-agent positioning mechanism in the dynamic environment", Springer Berlin Heidelberg, In RoboCup 2007: Robot Soccer World Cup XI ,2008.

[7] V. Monajjemi, A. Koochakzadeh, S. Shiry Ghidary, "grSim RoboCup Small Size Robot Soccer Simulator", Springer Berlin Heidelberg, 2011, pp. 450-460

[8] J. P. Mendoza, J. Biswas, P. Cooksey, R. Wang, D. Zhu, S. Klee and M. Veloso, "Selectively Reactive Coordination for a Team of Robot Soccer Champions", In Proceedings of AAAI-16, 2016.