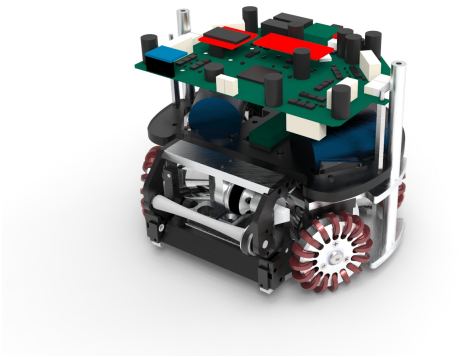
PARSIAN 2019
Extended Team Description Paper

Kian Behzad, Elham Daneshmand, Nadia Moradi, Mahdi Hajmohammadi Onidin, Mohammad Reza Kolani, Yasamin Alizadeh Gharib, Atiyeh Pirmoradi, Mohammad Mahdi Rahimi, Mohammad Mahdi Shirazi, and Mohammad Azam Khosravi

Electrical Engineering Department
Amirkabir Univ. Of Technology (Tehran Polytechnic)
424 Hafez Ave. Tehran, Iran
{kian.behzad,elham.daneshmand,nadiamoradi,mahdi.hmohammadi,mr_kolani,
alizadeh_yasi,,pirmoradi_atiyeh,mmrahimi,mhmmdshirazi,m.a.khosravi}@aut.ac.ir
http://www.parsianrobotics.aut.ac.ir

Abstract. This paper illustrates detailed mechanical, electronics, control, and software improvements made by Parsian Small Size Soccer team since last year. The notable mechanical change is improving the dribbler system to receive and control the ball more proficiently. Likewise, some improvements have been made in electronic to cooperate more efficiently with the software system. We used computational geometry algorithm to improve the path planning. Moreover, providing the inverse-model of the robots' kinematics was profitable to correct the robot motion. New developments in the *log analyzer* have also been provided. Lastly, an attempt to learn the opponent defense strategies have been explained.

Keywords: computer cluster, motion control, machine learning, opponent modelling



1 Introduction

Parsian, founded in 2005 by the Electrical Engineering Department of Amirkabir University of Technology, aims to design small size soccer robots compatible with International RoboCup competition rules. This team has been qualified for thirteen consequent years to participate in RoboCup Small Size League. First place in RoboCup2012 Passing and Shooting technical challenge, first place in RoboCup2013 Navigation technical challenge, and fourth place in RoboCup2012 and 2017 are the most significant achievements of Parsian team. Section 2.1 represents some mechanical changes in our robots. Then in section 2.2 electrical design features have been explained. In section 3 we discussed improvements in path planning and motion control. At last in section 4 new tools have been introduced in software.

2 Hardware

2.1 Mechanic

Changing the Robots Weight Distribution. This year, Parsian decided to change robots weight distribution completely. Parsian robots did not have a symmetrical center of mass because of the mass of the dribbler system in the front and the lack of weight balance in the back. This caused the robot to lose its balance in a sudden stop. To solve this problem, the team decided to move the battery position from the center to the back(Fig. 1 and 2). To implement this decision, the design of the middle plate of the robots has been completely changed, and for the battery, a plate was considered as a seat, and the battery has transferred from middle of the robot to the back. The new robots center of mass was checked with the CATIA and we made sure that it is closer to the robot's symmetrical axis. This has made the robot more stable.

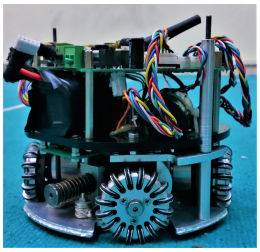


Fig. 1. The previous battery position

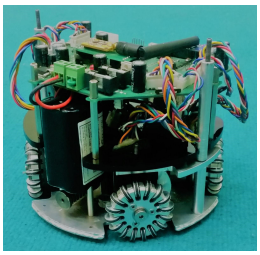


Fig. 2. The new battery position

Improved Ball Control Capabilities. When the ball was located in different positions in dribbler, it provided us with different speeds and direction in the same conditions. To resolve this problem the following changes were applied:

- **Curved Dribbling Bar**: The dribbling bar was redesigned with different curves and have tested with various covers. Although the ball could not be moved entirely to the middle of the dribbler system, the ball control was improved in a lateral motion.
- **Curved Kicker Head**: In order to shoot the ball in a straight line when the ball is at the corners of the dribbler the new curved kicker was designed. This caused the ball to move in a straight line when the ball is in the corners of the dribbler.
- **Side Appendages for Dribbling System**: In order to keep the ball in dribbler in lateral movements we designed the side appendages(Fig. 3). By applying force to the ball in lateral movements, these appendages retain the ball inside the dribbler and prevent separating it from the dribbler.

Receive Ball. One of the main challenges in the game is to receive the ball properly (with quick and accurate reaction). The damp system requires spring and damper to perfectly perform. To improve the efficiency of the damp system, a rotational axis were added to the dribbler system. Therefore, a new piece called "Cradle" (Fig. 4) was created that causes the circular movement of the damping system around the cradle axis. The result of the experiment shows that the cradle refined ball receiving.

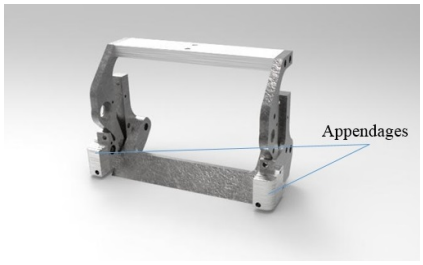


Fig. 3. The dribbler with side appendages

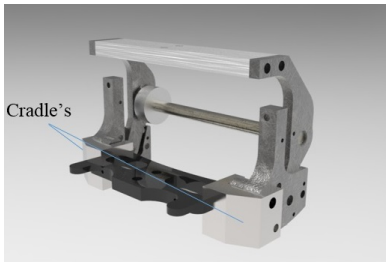


Fig. 4. The dribbler system with cradle

O-rings. silicon O-rings improved the coefficient of friction between the wheels and the ground (Fig. 5 and 6). In addition, the new O-rings resulted in smoother motion.

Fig. 5. The old wheel

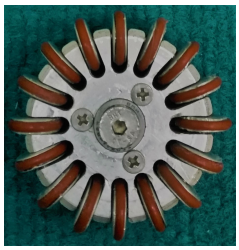


Fig. 6. The new wheel

Solenoid. According to equation 1, the ball velocity depends on the number of rounds of wire wrapped in kick coil due to the wire diameter and wire resistance and it has an optimum amount [1].

$$v(T) = \frac{\mu_{Fc} n A V_f}{4 R_{solenoid} m_p} T \tag{1}$$

The result of using 0.8 mm diameter wire and applying rounds of wire wrapped from 50 to 400 and 100, 150, 200 voltages was performed by MATLAB and was shown in Fig. 7, 8 and 9.

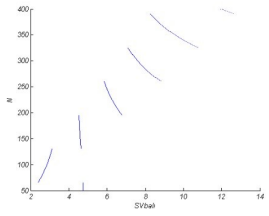


Fig. 7. 100V

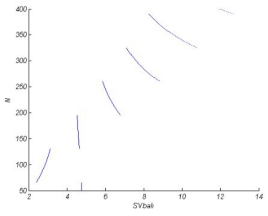


Fig. 8. 150V

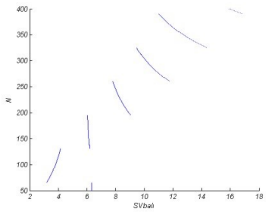


Fig. 9. 200V

Use Spring for the Kick Plunger. In order to make the robots more similar to each other, spring was used with the same stiffness factor for our kick plungers. (Fig. 10).

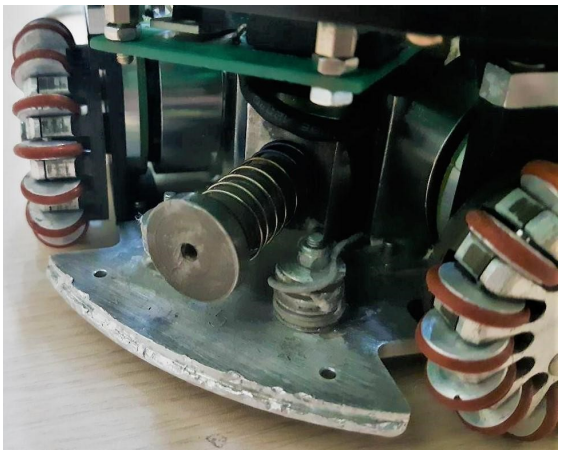


Fig. 10. Plunger with Spring

2.2 Electronic

Slip and Pushing Detection. Using the vision via Kalman, we capture the position, velocity, direction, and rotational velocity. Rotational velocity, linear velocities, and yaw angle are kinematic parameters provided through internal sensors. By comparing these two series of data, irregular motions such as slipping and pushing robot could be detected.

IMU Sensor Fusion. By using the rotational velocity from the gyro, the tilt angle from the accelerometer and compass of the magnetometer, and converting this information into the pitch, roll, and yaw angles, rollover of the robot could be detected. It shall be noted that the yaw angle calculated here is more accurate than the vision.

Active Controller Update. The parameters (PID coefficients) to control motor velocity, motor current, roll, pitch, and yaw angles can be updated with two-way communication. They are calculated and reported in real-time by *AI software* using two sets of data, vision and sensors output.

Debug Engine. This year Parsian has developed a debug engine which is used to debug the internal variables of the robot. The debug procedure starts from

the server request which sends the name of the internal variable. Afterward, the robots send the value of the requested variable to the server. (Fig. 11)

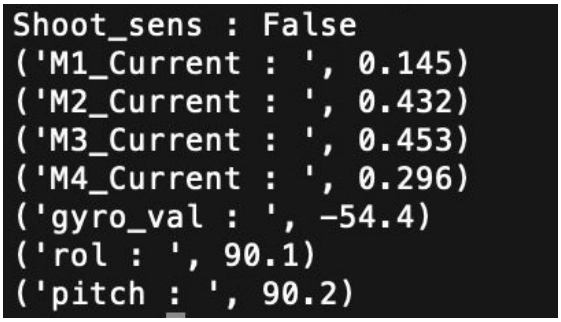


Fig. 11. reported variables

Future Design. The following considerations are planned to be taken into account for future:

1. Replacing Spartan 3 with Spartan 6.
2. PIC shall be substituted with Xmega since it is more frequently used in industrial application, more accuracy in ADC and it has less noise.
3. To have a single motor power switch, for each motor a MOSFET is used. Therefore, if a motor malfunctions, only that one will be disconnected and the rest of the motors continue working

3 Control

Computational Geometry Algorithm. For path planning, *ERRT* algorithm with a few changes has been used for several years in Parsian software [2]. The time complexity of this algorithm has grown exponentially with the increasing length of the path. Therefore, the *ERRT* algorithm was not performing sufficiently and consequently, a computational geometry algorithm was used. This new approach is called *GPlanner* which could be run in real time. *GPlanner* uses Tangent Visibility graph [3]. The new path planning algorithm brings about the following improvements:

1. *GPlanner* gives the shortest path.
2. The path is robust and each cycle does not change much.
3. It provides an easier way to use multi-agent systems.
4. *GPlanner* performed faster than *ERRT* on average.

Assuming the field as a rectangle polygon and each robot as a square, the center of which and robots are concentric, a polygon with convex holes representing the robots would be formed. Rohnert [4] has proposed an $O(n+h^2 \log h)$ time algorithm for computing the Tangent Visibility graph of P, where P contains h convex holes.
GPlanner computes the shortest path between two points inside a polygon P with holes of total n vertices can be computed in $O(n \log n + E)$ time with Dijkstra algorithm, where E is the number of edges in the Visibility graph of P. In each cycle, we assume a piece of the robot path as a rectangle [5]. It is an obstacle for the rest of robots and movable obstacles should correspond to convex polygons.

Learning-based Kinematics Correction. It was observed during the tests that robots tend to deviate from their trajectory when they are commanded to move only in one direction. Efforts have been put to correct such errors by online learning methods. In this application, a neural network is utilized in an inverse-model concept in order to estimate the relation between desired velocity and the actual velocity of robots.
The proposed network with "Adam" optimization algorithm to minimize the error of learning was applied and the update of weights would occur online, making it more flexible to be used in various environmental situations and be applied independently to every single robot as well. The block diagram of this workflow is shown in Fig. 12. Optimization is performed to minimize the difference between actual and observed velocities which is our cost function. This network has been tested several times with different structures of layers; though two-layer-network fitted best to the data. Apparently, three-layer-network results in the over fitting of the data which affects the online training procedure since the error for learning new data becomes larger.

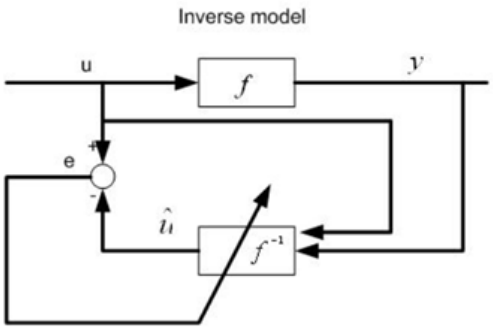


Fig. 12. diagram of inverse-model learning

Since forward, normal and angular velocities are dependent to each other (for example, by telling the robot to move only in forward direction, it may slightly move in the normal direction, too), the input and output of the network are the actual robot velocities and desired velocities, respectively. Therefore the inverse-model of the robots' kinematics was provided.

4 Software

According to Parsian TDP in 2018 [5], Parsian's code was moved to the Robot Operating System (ROS) framework. The experience from last year shows that using ROS has great advantages. Although running the nodes on one PC causes the shared memory protocol supersedes TCP [5] which was a great privilege compared to computer clusters. On the other hand, according to the new Rules in division A such as changing the field size and number of robots, running the code on one single system is hardly possible.

4.1 Computer Cluster

ROS is a distributed computing environment. A running ROS system can comprise dozens, even hundreds of nodes, spread across multiple machines [6]. Parsian stack is also designed with distributed computing in mind. Well-written nodes make no assumptions about where in the network it runs, allowing computation to be relocated at run-time to match the available resources [7]. Fig. 13 shows a running code on two machines with one agent on the field.

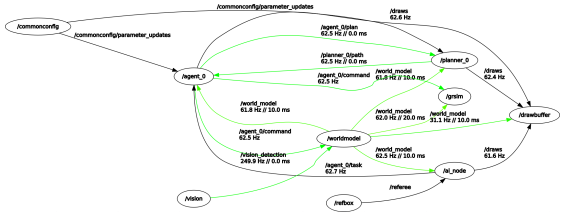


Fig. 13. Topics graph for one agent with AI in multiple machines

4.2 Software Architecture

Since last year, two minor changes have been made to the architecture of the Parsian Code. Both of these changes were aimed to make the testing and execution process quite easier.

Parsian−tools Package. The Parsian−tools package is the primary tool designed to allow the code to be implemented more quickly. This package provides several scripts in order to manipulate the software in different situations, also finding different tools in order to have a better and more reliable debugging process has become possible. Parsian−tools and its dependencies graph are shown in Fig. 14.

Parsian−sandbox Package. The parsian−sandbox package is a tool designed to perform quick and convenient tests of different behavioral actions. It has been implemented by the most used APIs in order to provide a better user experience. Parsian−sandbox and its dependencies graph are shown in Fig. 14.

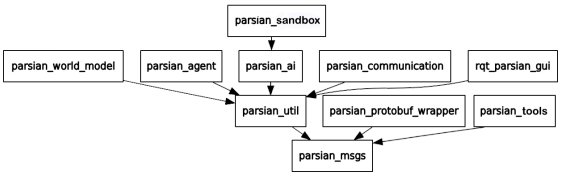


Fig. 14. Parsian packages graph

4.3 Log Analyser

The *log analyzer* was introduced in 2017 by parsian team [8] for the first time, this year it was expanded by developing new tools which are explained specifically in the following sections: Statistical Analyzer and Game Reporter. The first step towards analyzing games was to detect game events such as shot, pass, possession, and receive. An algorithm should distinguish them from each other during the match. For example, a robot kicks the ball potently to score a goal which increases the ball velocity immediately. This effect may occur while robots are passing to each other too. Therefore, It would be difficult to distinguish them from each other. Although using ball direction to differentiate them is common, having a partial viewpoint by the vision would cause problems; the chip ball path and ball collision are compelling instances of this. Therefore, it would be difficult to write a program considering all those situations. Since it would be hard to recognize the real events accurately, machine learning and classification methods are helpful. Consequently, It would be more accurate to determine the proper boundaries and separate event classes by decision-tree. The input data were analyzed logs which were labeled previously by Parsian members. A sequence of important factors grabbed from filtered vision data are given to the decision-tree to define the rules required for detecting game events. Ultimately, this program extracts a suitable knowledge that is beneficial in other tools that

are introduced in the following sections. It has been developed as a ROS node and using Scikit-learn library in python in order to learn the decision-tree.

Statistical Analyzer This program provides online statistical information about the game such as ball possession percentage, the number of yellow cards, shot succeed percentage, pass succeed percentage, ball possession heatmap, shot positions diagram, pass targets diagram and shot target diagram. Statistical data is the most important information for coaches in a real soccer match to make decisions. Likewise, the statistical knowledge presented by the *statistical analyzer* would help developers to detect bugs and weaknesses of their program. Game events and referee commands are the inputs for processing and calculating the statistical data. Ultimately, a GUI node will submit all statistics and data need to be drawn, then present and update them in an rqt plugin for each loop(Fig. 15).

Fig. 15. The shot analysis obtained from a part of a game: yellow robots show the successful ones and red robots represents the failed shots. It can be understood that the shots in a nearer distance to the goal are more probable to be succeeded.

Game Reporter The game reporter notifies the game events just like a real soccer match reporter. It utilizes the knowledge of game events and generates a proper sentence about each one and publishes them, which can be converted to speech by various developed ROS TTS tools.

4.4 Opponent Defense Modelling

Opponent modelling can help to predict opponent moves, perceive opponent team weaknesses, and simulate other teams' games. Our approach to model defense strategies uses extracted features from logs. Essential features fed as inputs are robots' position, velocity, acceleration, and direction. First of all, to make the dataset invariant to robots' id, we sort the robots of each team by their positions. To be more specific, we sort the robots based on two factors: the distance of each robot from their goal and the angle between the segment from robots position to their goal and the segment from their goal to the field center. Likewise, ball distance, ball angle (calculated same as we defined for robots), velocity, and acceleration was added to input features [9]. Moreover, we added symmetries of each row with respect to the horizontal and vertical axes crossing from the center of the field to our dataset in order to make it invariant to the sides of the field horizontally and vertically. Finally, we normalized all the data and prepared them for be learned by a neural network.

Firstly, assuming opponent robots located in the opponent half field as defense robots, we learned two-layered neural networks with 16 outputs which indicated the position of defense robots sorted by their distance to the opponent goal. As it has shown in the Fig. 16, the result was not sufficient.

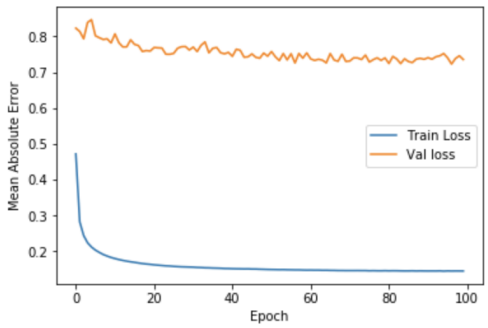


Fig. 16. Mean Absolute Error with respect to the epoch number

Secondly, we extracted the norm of the velocity vector for each defense robot to specify the reaction of them in different situations. The angle between the velocity norm vector and the vector connecting the filed center and right goal center is the final output of our model. This time, a neural network was implemented to learn that angle which indicates the behaviour of defense robots and goalkeeper. The result was compelling for the goalkeeper and the two nearest defense robots relative to the opponent goal, but we need to improve this method to predict the defense reactions more accurately. Another part of the game was used as the test dataset to assess this method; Fig. 17 represents the data loss function for both test and train dataset with respect to epoch number.

In the Fig. 18 and 19 the x-axis represents predicted values by the model and y-axis shows the true values for defense and goalkeeper. It is obvious that most of the points accumulate on the bisector for the goalkeeper which means that the predicted values and true values are accurately the same.

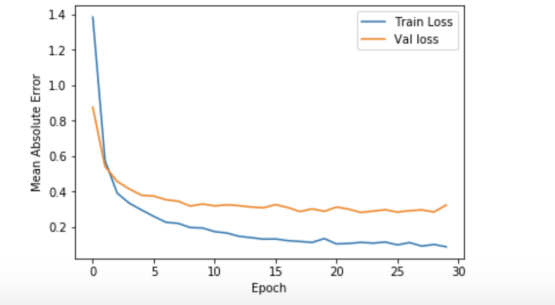


Fig. 17. Mean Absolute Error with respect to the epoch number

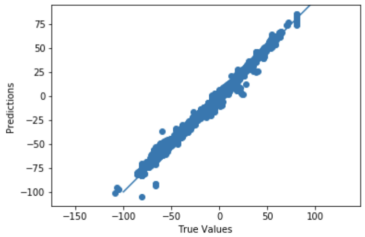


Fig. 18. Goalkeeper angle prediction: predicted values with respect to real values

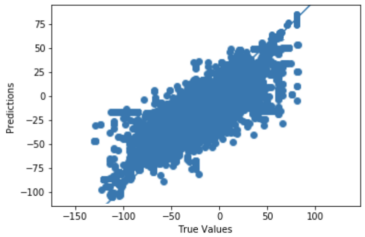


Fig. 19. Defense angle prediction: predicted values with respect to real values

5 Conclusion

This year our mechanical changes aimed to stabilize robots weight distribution and dribbler system. In the electrical section, some minor changes have been applied in order to reduce some systematic errors. In addition, some algorithms have been utilized to modify the path planning and velocity direction control. In the software section, all programming procedures could be distributed on multiple machines, also an analyzer tool has been introduced to report and analyze the games. More efforts needed to be put to achieve the ability to learn and predict the opponents defense during the game.

References

1. Naderi MA. Alireza Zolanvari, Mohammad Mahdi Shirazi, Seyede Parisa Dajkhosh, Amir Mohammad Naderi, Maziar Arfaee, Mohammad Behbooei, Hamidreza Kazemi Khoshkijari, Erfan Tazimi, Mohammad Mahdi Rahimi and Alireza Saeidi Shahrivar. Parsian 2015 Extended Team Description Paper for RoboCup. (2015)
2. Saeidi A., Malmir M.H., Shirazi M., Behbooei M., Boluki Sh., Kazemi M. and others, Parsian 2014 Extended Team Description Paper for RoboCup. (2014)
3. M. Pocchiola and G. Vegter. Minimal tangent visibility graphs. Computational Geometry: Theory and Applications, 6:303-314,1996.
4. H. Rohnert. Shortest paths in the plane with convex polygonal obstacles. Information Processing Letters, 23:71-76, 1986.
5. Rahimi, M.M., Shirazi, M.M., Gholyan, M.A.N., Chaleshtori, F.H., Moradi, N., Behzad, K., Roodabeh, S.H., Gavahi, A., Farokhi, F., Moghadam, S.A.G.A. and Gharib, Y.A., PARSIAN 2018 Extended Team Description Paper.
6. ROS/NetworkSetup - ROS Wiki, http://wiki.ros.org/ROS/NetworkSetup.
7. ROS/Tutorials/MultipleMachines - ROS Wiki, http://wiki.ros.org/ROS/Tutorials/MultipleMachines.
8. Rahimi, M.M., Shirazi, M.M., Arfaee, M., Gholian, M.A.N., Zamani, A.H., Hosseini, H., Chaleshtori, F.H., Moradi, N., Ahsani, A., Jafari, M. and Zahedi, A., PARSIAN 2017 Extended Team Description Paper.
9. Trevizan, F.W., Veloso, M.M.: Learning Opponents Strategies In the RoboCup Small Size League. In: Proceedings of AAMAS 2010 Workshop on Agents in Real-time and Dynamic Environments (2010)