PARSIAN 2019
Extended Team Description Paper

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 Khosravi

Electrical Engineering Department
Amirkabir Univ. Of Technology (Tehran Polytechnic)
424 Hafez Ave. Tehran, Iran
{kian.behzad,elham.daneshmand,nadiamoradi,mr_kolani,mahdi.hmohammadi,
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. The most significant achievements of Parsian team are first place in RoboCup2012 Passing and Shooting technical challenge, first place in RoboCup2013 Navigation technical challenge, and fourth place in RoboCup2012 and 2017. 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 software improvements have been introduced.

2 Hardware

2.1 Mechanic

Changing the Robots Weight Distribution. This year, the robots weight distribution changed completely. The Parsian robots did not have a symmetrical center of mass because the dribbler system is in the front side of the robot. This results in losing the balance of the robot in a sudden stop. To solve this problem, battery position changed from the center of the robot to the rear (Fig. 1 and 2). These changes transfer the center of mass of the robot from ($X = 0.297, Y = 55.589, Z = 36.803$) to ($X$=0.297, Y=49.26, Z=19.217); the center of mass has become 6.329mm closer to the bottom of the robot, and 17.586mm closer to the robot's symmetrical axis in the Z direction. A better performance was achieved since the robot's stability enhanced.

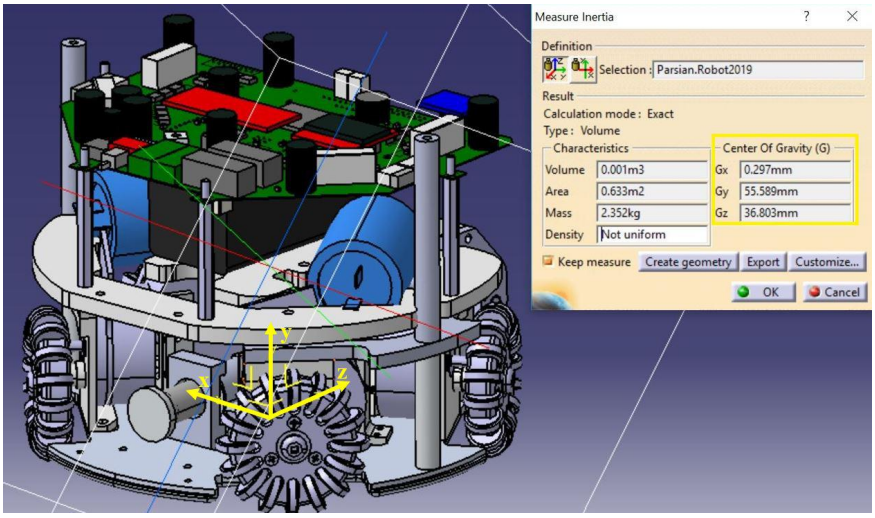

Fig. 1. The Previous battery position(the black cube represents the battery)

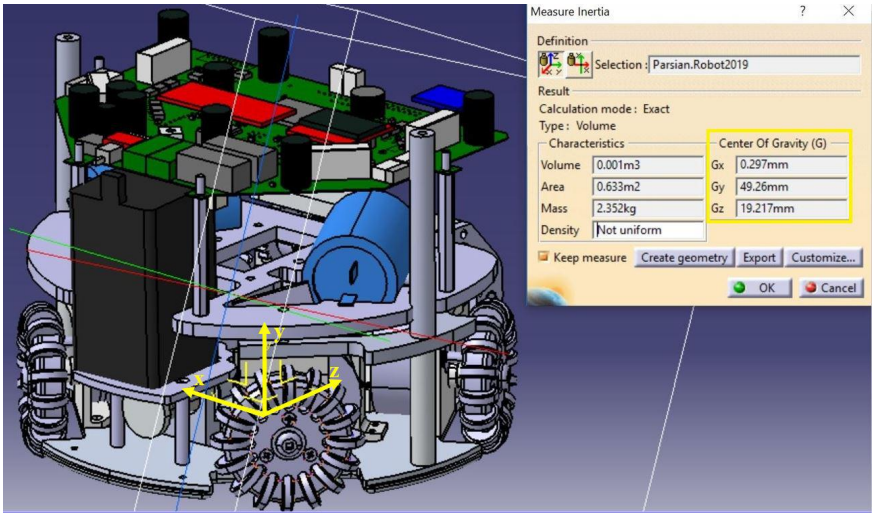

Fig. 2. The new battery position(the black cube represents the battery)

Improved Ball Control Capabilities. The dribbler and kicking system had a minor problem; when the ball was located in different positions in the dribbler, the ball speed and direction varied. Therefore, the kicking action was not

predictable. To solve this problem, the following changes have been applied:

– **Curved Dribbling Bar**: In order to bring the ball to the center of the dribbling system, The bar was tested with following specifications:
  - different curves of radius 350mm, 300mm, and 250mm
  - covers with a thickness of 2mm and 3mm

  Finally, the curve with a radius of 300mm (Fig. 3) and a cover with 3mm had the best performance based on the observations and experiments. Although the ball could not be placed exactly in the center of the dribbler system for the best result in control, the ball control was improved in the lateral motion.

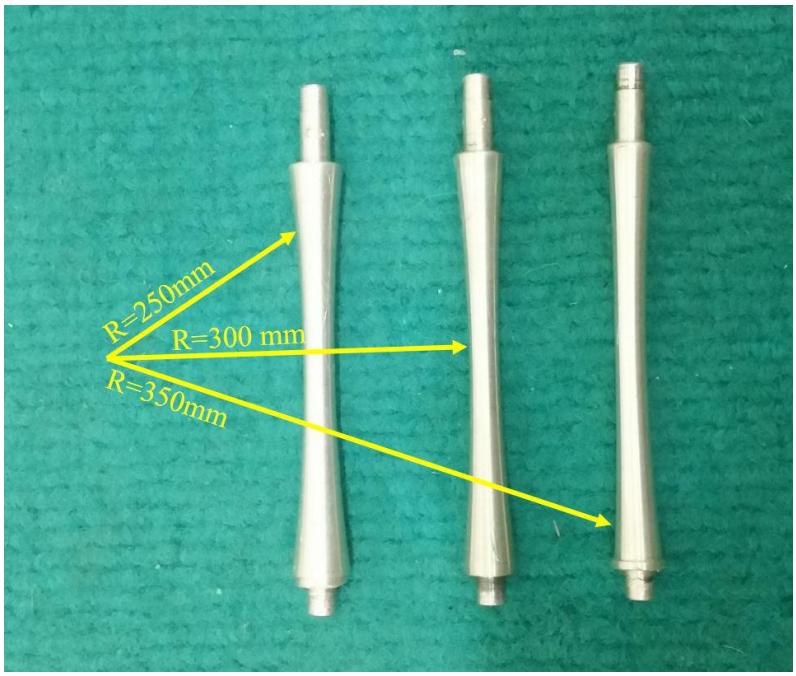

Fig. 3. curve dribbling bar

– **Curved Kicker Head**: When the ball is at the corners of the dribbler so as to shoot the ball in a straight line, a new curved kicker of the radius 500mm, 400mm, and 300mm tested to lead the ball to the center-line of the curved kicker. The curve kicker with 400mm (Fig. 4) radius gave us the best performance based on observations. In a radius of 500mm, the effect of the

curve is not considerable. On the other hand, the radius of 300mm causes a large deviation of the ball. We figured out that the 400mm radius curve is the most suitable one. This caused the ball to move in a straight line when the ball is in the corners of the dribbler.

Fig. 4. curve kicker bar

– **Side Appendages for Dribbling System**: In order to keep the ball in the dribbler in lateral movements, we designed the side appendages (Fig. 5). By applying force to the ball in lateral movements, these appendages retain the ball inside the dribbler and prevent it from separating 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, another rotational axis was added to the dribbler system. Therefore, a new piece called "Cradle" (Fig. 6) was created that causes the circular movement of the damping system around the cradle axis(Fig 7 and 8).

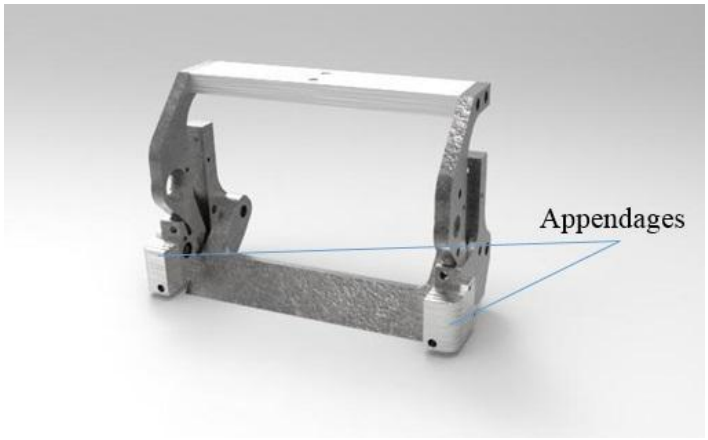

Fig. 5. The dribbler with side appendages

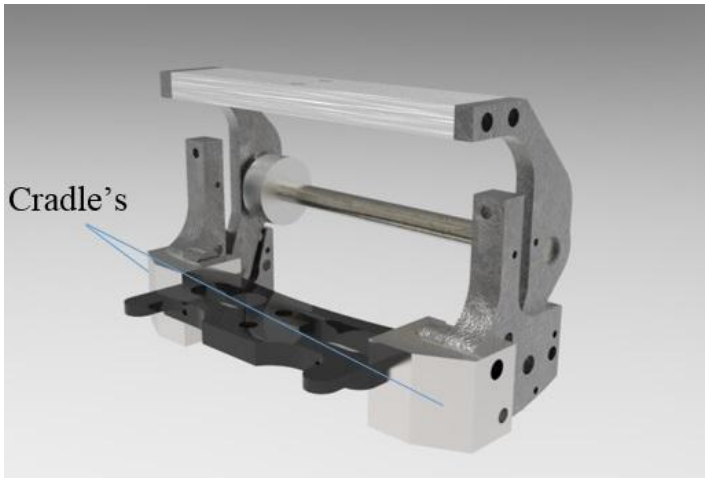

Fig. 6. The dribbler system with cradle

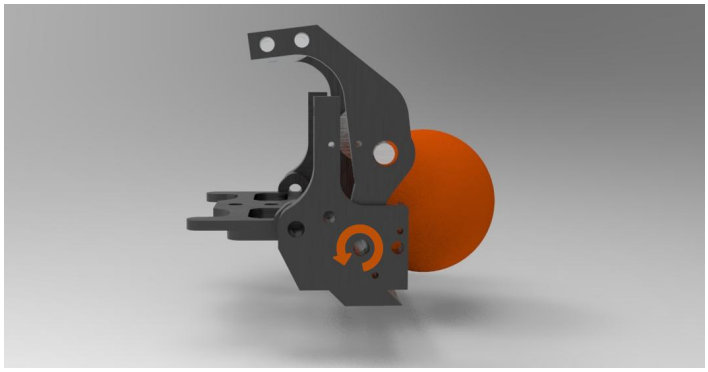

Fig. 7. The old dribbler with one rotational axis

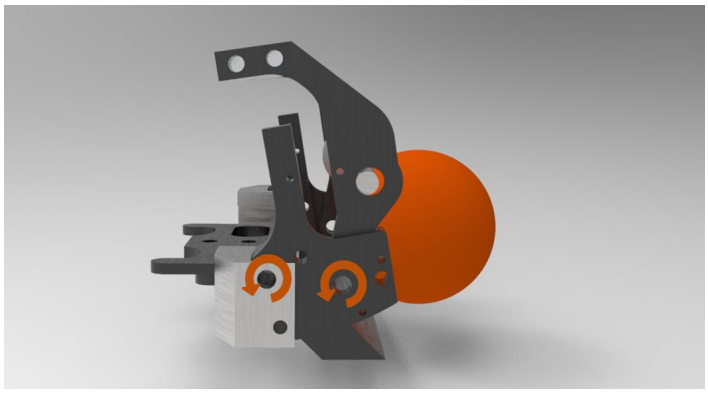

Fig. 8. The new dribbler with two rotational axis

O-rings. Compared to the previous plastic O-rings, silicon O-rings increased the coefficient of friction between the wheels and the ground(Fig. 9 and 10); it results in a smoother movement of the robot.

Fig. 9. The old wheel

Fig. 10. The new wheel

Solenoid. According to the equation 1, 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 value [1].

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

Results of using 0.8 mm diameter wire and applying rounds of wire wrapped from 50 to 400 and 100, 150, 200 voltages were shown in Fig. 11, 12 and 13. In which the horizontal axis, SVball, indicates speed of ball after kicking, and the vertical axis, N, states the number of coil wire turn.

The jumps in the diagram are due to the change in the coil diameter of the time to reach the last and first winding path.

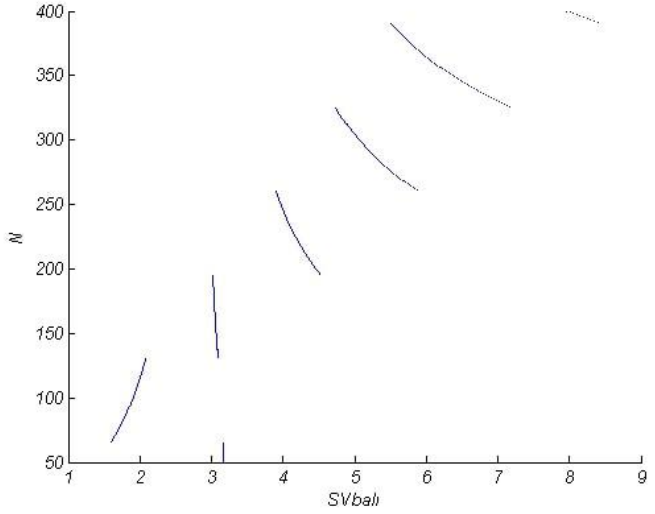

Fig. 11. 100V

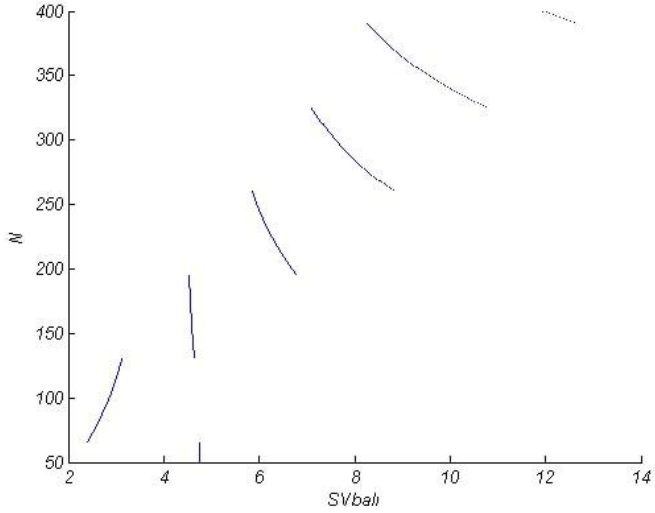

Fig. 12. 150V

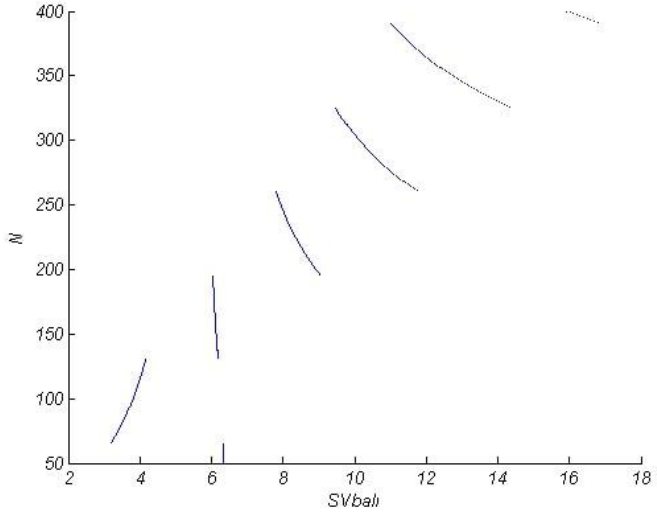

Fig. 13. 200V

Use Spring for the Kick Plunger. To have similar properties in all robots, similar springs are used to have the same stiffness factor for all kick plungers. (Fig. 14).

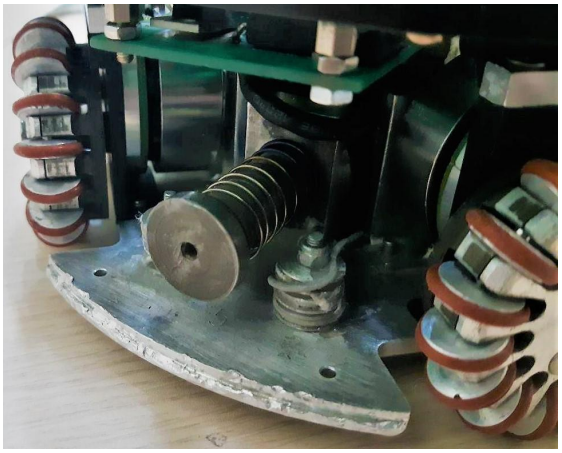

Fig. 14. Plunger with Spring

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. 15)

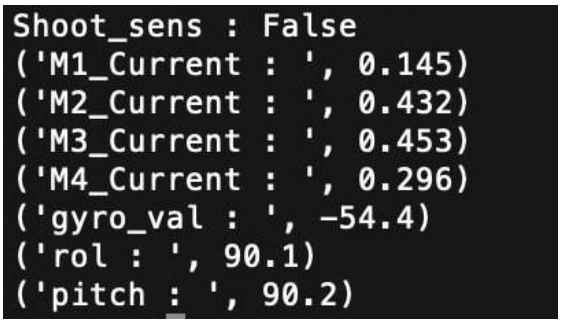

Fig. 15. reported variables

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 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. The block diagram of this workflow is shown in Fig. 16. Optimization is performed to minimize the absolute of difference between

actual and observed velocities which is our cost function. This network has been tested several times with different structures of layers; though a fully-connected two-layer-network fitted best to the data using 100 neurons in each layer and ReLU as activation function for each neuron. Three-layer-network results in the over fitting of the data which affects the training procedure since the error for learning new data becomes larger.

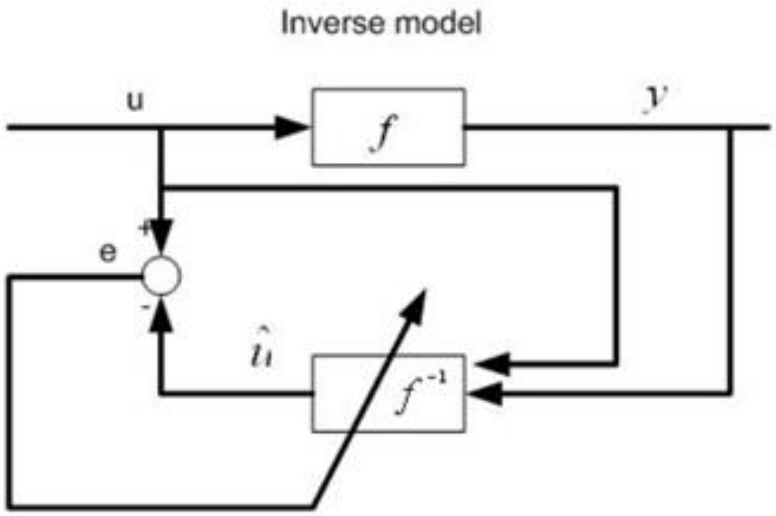

Fig. 16. diagram of inverse-model learning

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 desired robot velocities and actual velocities, respectively. Therefore the inverse-model of the robots' kinematics was provided. In order to use the network after training, a set of desired velocities is fed to network and The output of network (the velocities needed to be published for the robot to move according to the desired velocities), is calculated. This output is used continuously to correct the motion of robots(Fig. 17).

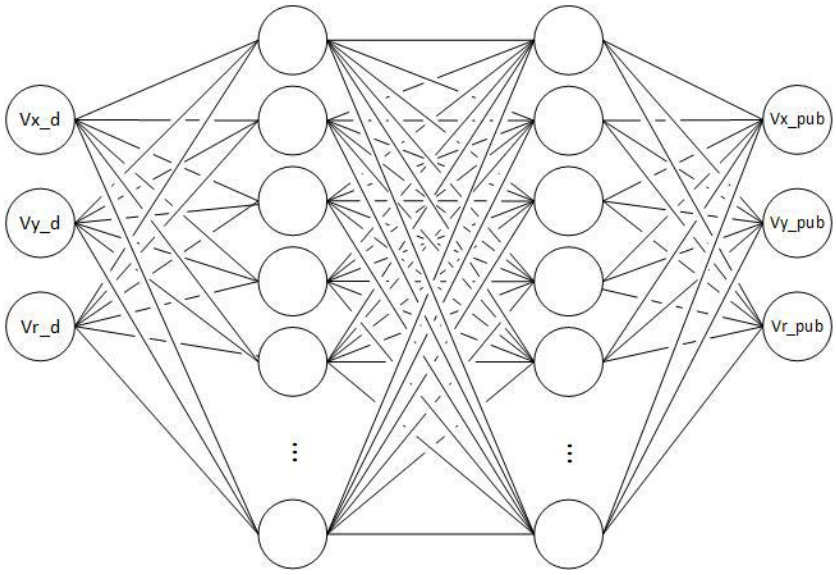

Fig. 17. schematic representation of a two-layer fully-connected Neural Network

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. 18 shows a running code on two machines with one agent on the field.

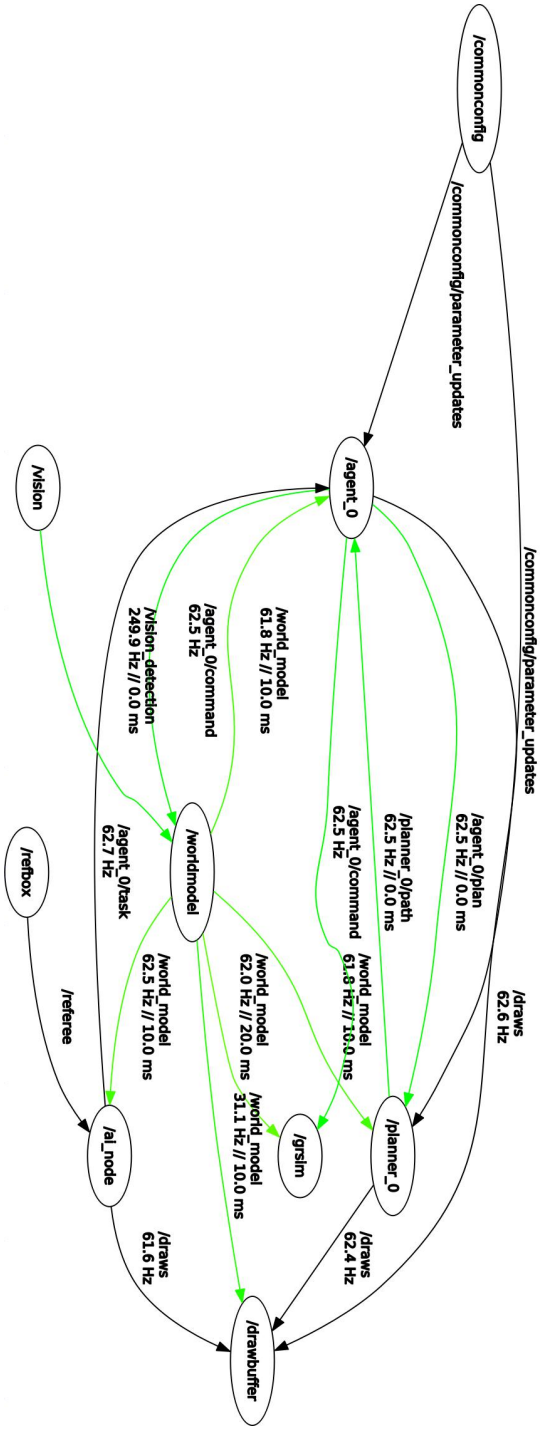

Fig. 18. Topics graph for one agent with AI in multiple machines

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. 19.

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. 19.

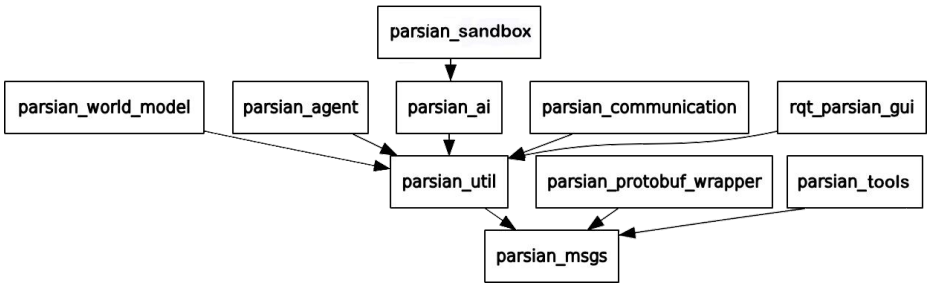

Fig. 19. Parsian packages graph

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. 20).

Fig. 20. 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 be helpful to predict opponent moves, perceive opponent team weaknesses, and simulate other teams' strategies. An approach to model defense strategies is to use past logs and extract features from them in order to learn the opponent strategies. Variables of robots and the ball including position, velocity, acceleration, and direction are effective features for learning. Therefore, these features are extracted from logs and then preprocessed to be fed as inputs to the learning method. As the first stage of preprocessing, we defined two new features to be used instead of position:

1- the distance of their goal
2- the angle between the segment crossing from its position to their goal and the segment from their goal to the field center.

Moreover, to make robots invariant to robots' id, the robots of each team are sorted by these two features. In addition, in order to make the data-set invariant to the sides of the field horizontally and vertically, symmetries of each row with respect to the horizontal and vertical axes crossing from the field center were added. Finally, we normalized the data and fed them as input to the neural networks [9].

After extracting useful features from logs for each frame, considering a specific team as opponent, we tried to learn their defense strategies. Assuming opponent robots located in the opponent half field as defense robot and filling out other opponent robots' position in the data-set with zero, we tried two methods that are explored in the following:
Firstly, we fed gathered data for one team and the ball as input. The two features defined as positions for 8 robots of the opponent team are considered as outputs. We tried to learn two-layered neural networks with 16 outputs which indicated the number and position of defense robots. As shown in Fig. 21, the result was not sufficient.

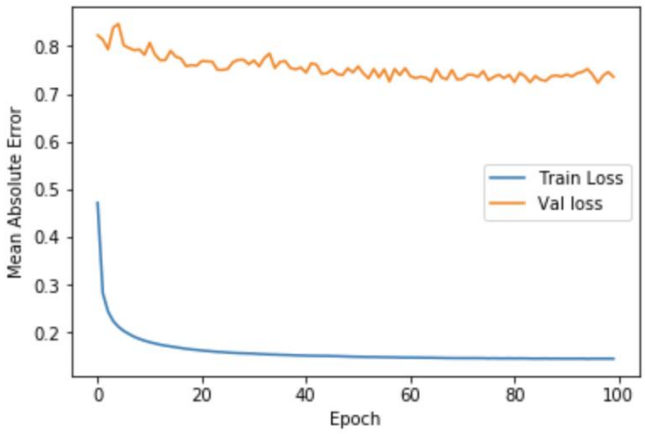

Fig. 21. Mean Absolute Error with respect to the epoch number

Secondly, we extracted the norm of the velocity vector for each defense robot to learn 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 behavior 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. 22 represents the data loss function for both test and train dataset with respect to the epoch number.

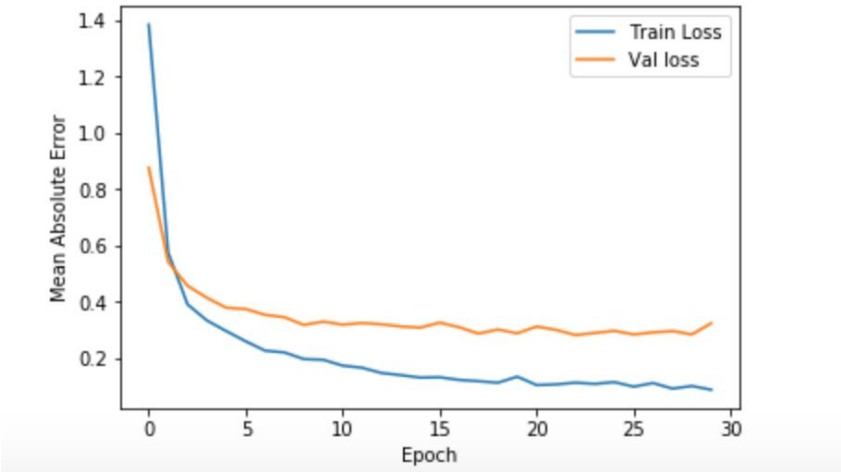

Fig. 22. Mean Absolute Error with respect to the epoch number

In Fig. 23 and 24 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. Still, need to test other learning methods such as deep learning to get better results.

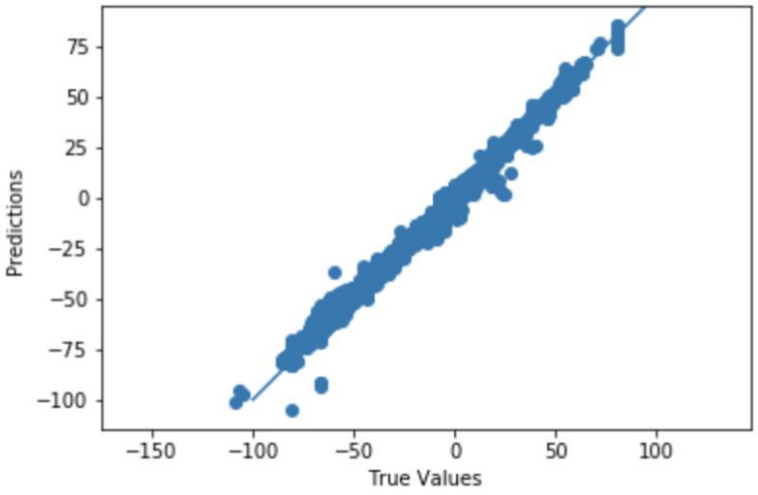

Fig. 23. Goalkeeper angle prediction: predicted values with respect to real values

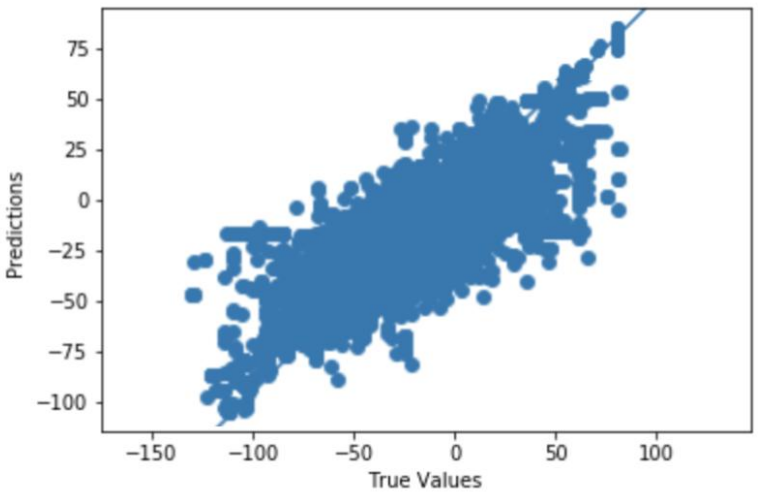

Fig. 24. Defense angle prediction: predicted values with respect to real values

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)