CRONE 2018
Conference Proceedings

AIS: Artificial Intelligent Soccer

Ricardo Alfaro[1*], Maximiliano Aubel[1*], Pablo Yañez[1*], Pablo Reyes[1], Tomás Rodenas[1], Nicolás Hernandez[1], Felipe Pinto[1], Sung Hee Kim[1], Tania Barrera[1], Daniel Torres[1], Ignacio Vicencio[1], Jorge Alvarez[1], Felipe Osorio[1], Sebastian Castillo[1]

***For correspondence:**
ricardo.alfaro.13@sansano.usm.cl; maximiliano.aubel@sansano.usm.cl; pablo.yanez.12@sansano.usm.cl

[1]Innovación y Robótica Estudiantil, Universidad Técnica Federico Santa María

Abstract This paper describes the current development status of our SSL team, AIS. Throughout this document, we present the design and implementation we have got so far, showing the electrical, mechanical and software topics involved in our work, which were designed according to satisfy the RoboCup rules. In addition of giving details on the current development, we present some insights on the main challenges that have been identified and tackled in recent years with the consolidation of this team, in order to foster other forming groups around the globe.

Introduction

Innovación y Robótica Estudiantil, which is the affiliation of all members on this team, has been founded in 2001 and corresponds to a self-organized group of students from several faculties, such as Electronics, Informatics and Mechanical Engineering departments at the University (UTFSM). This RoboCup team belongs to one of several projects from this aggroupation and is conformed by students with different specialization areas such as computer science, control, and automation, or power electronics, but also on a multidisciplinary approach including students from mechanical engineering, as well as industrial engineering students.

This SSL team follows the nature of the host students initiative, starting from its multidisciplinary constitution, the self-organization and motivation with professor advises when required but managed independently from any professor funding project, and trascendence over generations renewing its members with a constantly growing development and enhancement, and making both research and development works, like [1] where a previous generation of the team applied reinforcement learning on the goalkeeper task.

This document describes our design and the implementation we have got so far, showing all the work made in the different areas involved in this category.

In particular, we describe the mechanical design, electronics design for different devices and algorithms implementation for the (robotics) team coordination, also including the expected implementation we are planning to reach by the time of the competition.

Mechanical Design

The material selected for the chassis structure is chosen by means of priorizing the collision resistance, so an aluminum base is used, while supports for the wheel motors also consists on four aluminum blocks and a second floor of polymethyl-methacrylate (PMMA) which stands for supporting the battery. Then, a third floor is designed also of PMMA, with the aim of supporting the PCB and also isolating the battery and PCB. Finally, the cover is 3D-printed on ABS material.

-Height: 150 mm.

-Diameter: 180 mm.

-Maximum coverage of the ball: 18%.

Figure 1. Omnidirectional wheel back view

Figure 2. Robot assembled

Drive System

Mechanical locomotion of robots is based on 4 omnidirectional wheels, which are currently 3D-printed in PLA. Each wheel is designed with 55 mm of diameter and 15 sub-wheels of 13.5 mm of diameter, so the robot can move in all directions. Also, each one of the 55 mm diameter wheel has a set of 15 mini metal V grove guide pulley rail ball bearing wheels. Each wheel is driven by a Maxon EC45 30 Watts motor [7] and an L6235 driver for 3-phase brush-less DC motor [6], which enables us to program a velocity control for each motor, ensuring that the robot moves to our desired setpoint speed.

Hardware

Each robot is controlled by a PIC MX440F256H using Pinguino Development board. This model was chosen because of its simplicity, versatility and peripherals features. It has shown an acceptable performance, letting us accomplish communication, movement, and playing skills. The peripherals also replace a lot of external electronics needed to control the motors and dribbler.

Peripherals

ADC conversion

The ease of implementation of this kind of modules allows us to control wheel speed and orientation through an L6235 driver using DAC conversion. Additionally using ADC conversion we can measure the wheel speed, allowing us to implement a PID control on velocities for each wheel.

I2C

As mentioned before, we use an L6235 driver, which communicates with the PIC through its I^2C module.

UART

The UART module allows us to develop serial communication between the PIC and our APC220 RF module, which will send and receive data from the centralized decision maker placed on the computer. Additionally, we use an FTDI connected to our UART communication module to watch data of interest.

GPIO

The general purpose Input/Output pins let us program easily, in general, any other significant settings, i.e. set the wheel break and direction pins or activate the kick routine.

Kicker

The circuit that is shown in Figure 3 is used for the kicking system. This consists of a chip charger controller with regulation which is a controller of flyback of high voltage, raising the voltage from 24 [V] to 100 [V] on a capacitor of 2400uF and, therefore, storing an energy of 244 [J]. The time of charge of the capacitor takes up to 3 seconds to reach the voltage setpoint and can be regulated to kick with different intensities.

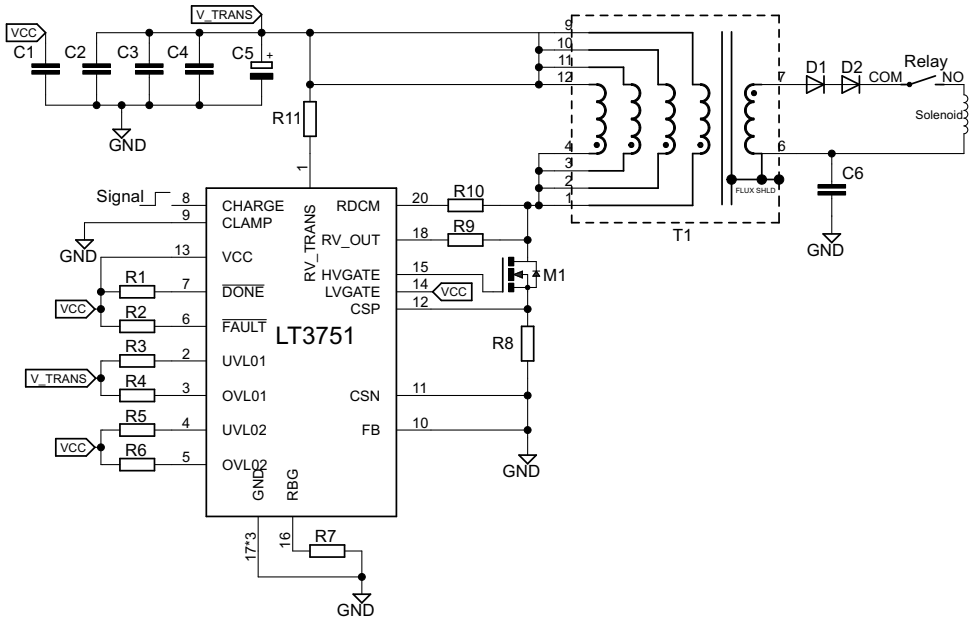

Figure 3. Kicker circuit

Dribbler

According to RoboCup rules, the robot is allowed to cover up to 20% of the ball. Experimentally, it has been proved that it is easier to catch the ball when the dribbler has a slight curve to center the ball on its own. So, this design involves two diameters, D1 and D2, and based on this information, maximum height possible is calculated obtaining the following expression:

$$H = \sqrt{\frac{1}{4}(D_2(2d + D_2) + D1(4pd - 2d - D_1) + 4pd^2(1 - p))} + \frac{d}{2}, \quad (1)$$

where d and p corresponds to the ball diameter and maximum coverage of the ball, whose relation is illustrated in Figure 4.

Our team uses the engine MAXON EC 16, BRUSHLESS, 30 WATT, SENSORLESS, handled by an ESC LettleBee opto 6s.

Both the engine and roller join using a gear system with ratio 1:1, configuration that let us drive

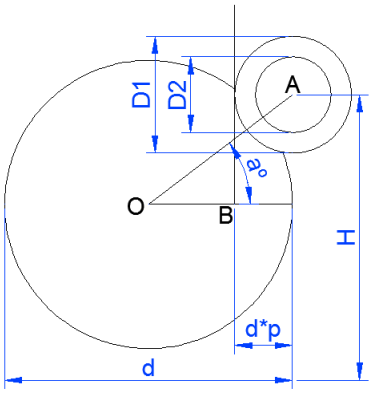

Figure 4. Relation between variables involved in dribbler design

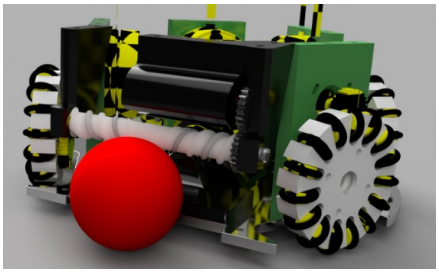

Figure 5. View of dribbler assembly

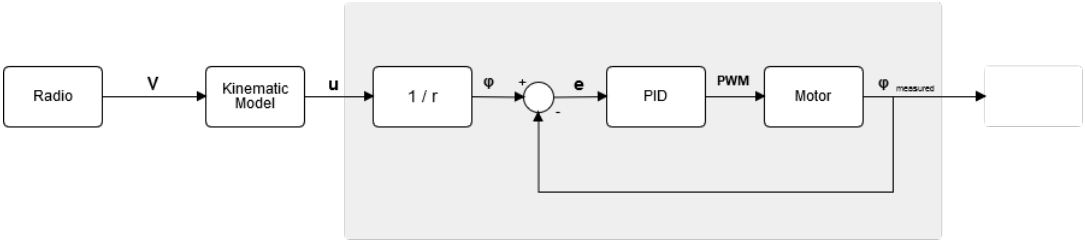

Figure 6. Robot wheel speed control system

and automatically center the ball with a 3D-printed support structure.

Communication

For communication, we use an RF module consisting of an APC220 which is a low-cost NRF Athena that integrates an embedded high-speed microprocessor and high-performance IC that creates a transparent UART/TTL interface. It is a 430 MHz system capable of transmitting up to 1000 meters. We send every single robot data through a common channel as hexadecimal packages in order to achieve better transmission bytes. Each robot receives and decodes the data in a pic microcontroller.

Kinematic model and wheel speed control

In order to maintain the expected velocity and position, we applied PID control [2] on every wheel once the setpoint speed is calculated for every robot. To accomplish this task, our control system sends a velocity vector $V = [v_x, v_y, v_\theta]^T$ to each robot, multiplying then its kinematic model matrix W, defined by the geometry of the robots, obtaining

$$u = V^T W = [u_1, u_2, u_3, u_4]^T = \phi r,$$

where u is the wheel velocity vector, which divided by r (radius of the wheel), it is possible to obtain the angular velocities of the wheels, ϕ. This is the reference variable to the control speed system shown in Figure 6, and by obtaining direct measure from the hall sensors of the motor we can obtain the input error variable to the PID. Then, the controller generates a PWM as output in order to set the wheel speed. It is important to note that ϕ is treated as an absolute value because the direction is set by enabling or disabling certain pins on the driver controller.

Software

Diagram depicted in Figure 7 shows a general overview of the system, where we implement a high-level AI decision making in order to decide which is the best action to take from a set of high level preprogrammed plays based on the game state that comes just from the vision receiver. Then we have a low-level path planning algorithm to choose the best path in order to execute the play avoiding obstacles. This is implemented on the desktop computer in charge of making the centralized decisions for every robot.

High level AI

The higher order AI level computes at each processing cycle the best actions to be performed for each robot. This action is chosen by selecting a game-play from a pre-defined static pool. This fundamental part of the system's architecture is shown in Figure 8, introducing four identifiable processes. First, there is a *SceneRater* which analyze and encapsulates all the relevant information from the game field for choosing a game-play. Then, that information is used for actually selecting the specific game-play through the block named as *PlayChooser*, weighting each detected event for deciding whether an attacking strategy or a defensive one should be used, and which one in detail

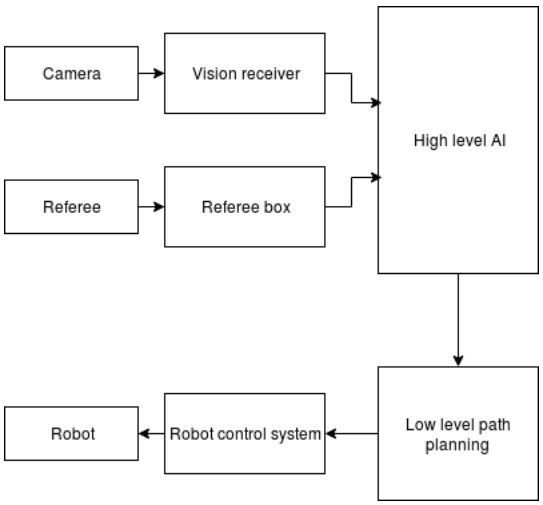

Figure 7. General diagram of the system

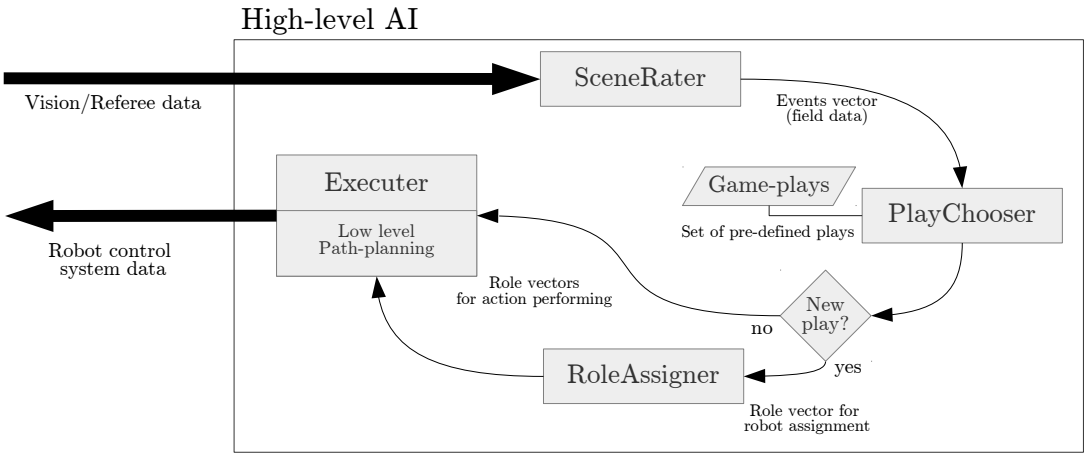

Figure 8. High-Level Artificial Intelligence Architecture Diagram

must be performed. Once a game-play is chosen, then a *RoleAssigner* block is in charge of coherently distribute the roles associated to that game-play, as well as selecting which robots should assume each position. Finally, each position must be run by the robots, a process managed through time by an Executer block.

Then, as shown in Figure 8 which depicts the diagram of the algorithm implemented as the top hierarchy intelligence architecture, the processing cycle starts with the receiving of new data either from the vision system or the referee. As illustrated, four blocks are implemented and processed in order: SceneRater, PlayChooser, RoleAssigner, which is executed just if the current play has changed, and the Executer block.

Specifically, the *SceneRater* evaluates conditions as which team has the ball, whether a team has or not high and middle chances of making an annotation, the partial position of the ball in the field, which team is closer to the ball, among others relevant features.

The *PlayChooser*, after receiving the vector of detected events, evaluates all pre-defined game-plays, each one of them rating differently the events detected. Offensive plays rate with higher values the detection of the ball in the enemy team area, and even more if the ball is close to the enemy goal area. Coherently, defensive plays strongly rate when the enemy team has the ball and even more if they have an opportunity for shooting to the team goal area.

Each game-play is described by a set of roles, one role for each agent, introduced in a priority

order in case of using fewer robots than the maximum allowed. Each role considers a set of actions to be developed by the agent, as moving, receiving or giving a pass or shooting to the goal area.

To do so, the architecture includes the *Executer* block, which is in charge of evaluating if either an action has finished or not, managing the changing of steps and computing the next step of an action execution for each robot.

In order to simulate the robotic team coordination, and test different multi-agent algorithms, we make use of GrSim [5], software that has been very helpful to test game strategies.

Low-level path planning

Under the high-level plays, we run a path planning algorithm to find the best way of executing these tasks. We have tested different methods looking for a suitable algorithm which gives good results at the moment of avoiding obstacles.

The first method tested was Potential Field algorithms [4]. This proposes a potential field representation for obstacles and target, using sources for the prior and sink for the latter. In this way, vector trajectories are generated avoiding obstacles and leading the agent to the target, as we let a ball fall down. A disadvantage of this method is that we could obtain local minimums without reaching the target.

The best method tested was Rapidly-Exploring Random Trees (RRTs)which consists on expanding a tree on the target zone, avoiding to add nodes that could produce collisions with targets. The added points to the tree are randomly chosen with probability p in a straight line to the target, and with probability $(1 - p)$ selecting a random point on the space, making more exploration and avoiding to get stuck on a different location to the target.

For improving its performance, we have implemented and tested some of the algorithms based on RRT, way-points, smoothing and some extensions like RRT* presented on 2011 [3].

Conclusions

Although we tackled several challenges prior to our participation at the Robot Soccer World Cup 2018, such as discarding low-level path planning algorithms due to local optimums according to some remarks described throughout this document, and despite efforts about testing game strategies on a simulated environment, a newcomer team into the league should consider several other challenges that should be faced at the time of participation, such as:

- Full integration with referee box: it is not enough to communicate this software with the central unit for the team decision-making system, given that all rule cases should be covered.
- Number of robots: although each team is allowed to participate with a smaller number of robots, it is important to maximize the number of available prototypes in case of structural damage (chassis should be prepared to receive very strong shoots).
- Number of team members: in case of belonging to a self-organized group of students (such as our case), budget and funding search for the tournament participation should include at the very least six members, given that captain is constantly being called for meetings within the contest, and two other members have to serve as referees for other matches.
- Precision and time delays: although splitting of teams into Division A and Division B lower the game complexity in order to face similar teams in terms of experience and tasks achievement, there are very experimented teams which have mastered the motion control of robots, so it is important to be prepared to chase and face the ball in the most efficient possible manner.

Next challenges for this team in order to qualify for the upcoming world cup in addition to fundraising for participation, include fine-tuning of robots which were already capable of playing matches and even winning one of them, following with the appropriate prototypes replication.

References

[1] Ahumada et al. “Accelerating Q-Learning through Kalman Filter Estimations Applied in a RoboCup SSL Simulation”. In: *Robotics Symposium and Competition (LARS/LARC), 2013 Latin American*. IEEE. 2013, pp. 112–117.

[2] Åström et al. *PID controllers: theory, design, and tuning*. Vol. 2. Instrument society of America Research Triangle Park, NC, 1995.

[3] Bry et al. “Rapidly-exploring random belief trees for motion planning under uncertainty”. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 723–730.

[4] Ge et al. “New potential functions for mobile robot path planning”. In: *IEEE Transactions on robotics and automation* 16.5 (2000), pp. 615–620.

[5] Monajjemi et al. “grsim–robocup small size robot soccer simulator”. In: *Robot Soccer World Cup*. Springer. 2011, pp. 450–460.

[6] Vincenzo Marano. “L6235 three phase brushless dc motor driver”. In: *Application Note, ST* (2003).

[7] AG Maxon Motor. *EC-Powermax 30 Catalogue Information*. 2008.