ZJUNlict Extended Team Description Paper

Small Size League of Robocup 2024

Anke Zhao, Pengfei Yu, Zheyuan Huang, Ning Shen, Jialei Yang, Jiazheng Yu, Zhike Chen, Liang Wang, and Rong Xiong

State Key Lab. of Industrial Control Technology
Zhejiang University
Zheda Road No.38, Hangzhou
Zhejiang Province, P.R.China
rxiong@iipc.zju.edu.cn

Abstract. This paper mainly describes the work of the ZJUNlict team in the past year in both hardware and software. In the hardware part, new control and communication architectures were designed based on v2023 robot [1]. In the software part, decision module was reorganized in a modular way, and attacking and defending strategies were modified to adapt to the high-tempo games, including a brand new defence system. A standard benchmark was also proposed for teams to evaluate the capabilities of their robots in an objective way.

1 Introduction

ZJUNlict is a team that have been participating in the competition since 2004. In 2023, we returned to offline world competition after the pandemic and achieved the second place, demonstrating our great efforts during the years. We've made several innovations and improvements both in hardware and software in the past year, which will be presented in detail in this paper.

The improvements we've made in hardware based on our newly developed v2023 robot system will be introduced in Section 2, where we explain the working principle and specific implementation behind embedded multithread control and wifi dual-path communication.

Section 3 involves our work in the software. We redesigned our decision module to organize all the improvements we've made since 2019 in a modular way for better extensibility. With the experience from our first 11 vs 11 world competition, we've also designed new attacking and defending algorithms. The new off-ball running and dual-scoring passing evaluation as well as immediate passing and shooting judgement aims at improving the efficiency of attack, while the new defence system is targeted at a better task dispatching with 11 robots.

ZJUNlict team has always been trying to contributing to the SSL community in various ways. As we designed a new version of robot, we came up with the idea of a standard benchmark to evaluate the capabilities of robots. We'll explain the item of the benchmark in detail and share the result of our v2019 [3] and v2023 robots in Section 4.

2 Hardware

In 2023, a new robot system was introduced and its hardware design was documented in the ETDP [1]. Following extensive improvements and verification over the course of more than half a year, the v2023 robot system has demonstrated robust reliability. Notably, significant advancements have been achieved in the embedded and WiFi communication subsystems, surpassing the capabilities of the v2019 robot system [3].

2.1 Multithread Control

In the v2023 system, the main control unit was upgraded to the Raspberry Pi CM4 module. This upgrade offers several advantages, including a higher operating frequency limit and the ability to support multi-threaded control.

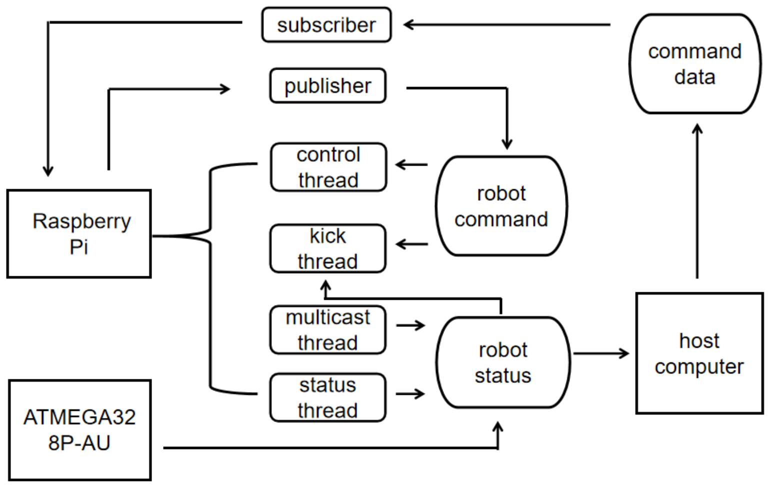


Fig. 1: Multithread structure on v2023 embedded system. The embedded program for the v2023 robot has four threads that share resources for robot command and robot status via thread locks. Subscriber and publisher control the command data from the host to the Raspberry Pi CM4 and the updating of robot command resources by the Raspberry Pi CM4, respectively.

In v2019 robot system, STM32H743ZIT6 was utilized as the main control unit, which operates at a maximum frequency of 400MHz. When it comes to the v2023 robot, it was upgraded to the Raspberry Pi CM4 module, which allows the robot to achieve a maximum operating frequency of 1.5GHz. This upgrade has greatly enhanced the response speed of the robot system.

With the increased processing power of the Raspberry Pi, implementing multi-threading to handle various tasks becomes possible, replacing the previous interrupt-based approach.

The main control functionality is now divided into four threads to enhance efficiency, as shown in Fig 1. Thread one is responsible for controlling the robot's movement and ball dribbling. Thread two handles the transmission of the robot's status data packets back to the host computer. Thread three establishes communication with the host computer using UDP through multicast. Thread four controls the robot's shoot-chip action. The tasks performed in threads two and three are closely tied to the robot's communication mechanism, which will be elaborated on in Section 2.2.

Additionally, the instruction packet from the host computer, the robot's state status, and the robot's command status serve as shared resources, enabling seamless access and collaborative work among the different threads. To prevent data races arising from multiple threads accessing shared resources simultaneously, thread locks have been implemented for two specific shared resources, "robot command" and "robot status". When a thread attempts to access a resource, it must acquire the corresponding mutex. If the mutex is currently unlocked, the thread will successfully obtain the lock, enabling safe access to the resource. If the mutex has already been locked by another thread, the current thread will need to wait until the mutex is unlocked.

2.2 Dual-path Communication Based on WiFi6

In the v2023 system, "dual-path communication" was implemented using WiFi6, combining both unicast and multicast transmission. This approach ensures that the communication system has ample bandwidth to accommodate the requirements of up to 11 robots operating simultaneously on the same stage, as per the new rules introduced in 2021. Additionally, this configuration effectively controls the packet loss rate even in such demanding scenarios.

In v2019 communication system, NORDIC's nRF24L01 product, a highly integrated 2.4GHz wireless ISM(Industrial Scientific Medical) band transceiver chip, was utilized. The communication subsystem of the v2019 robot has consistently demonstrated stability, making it a reliable foundation for our high efficient communication protocol. [3].However, over the past few years of using this system, some certain limitations are increasingly worthy of attention.

The communication rate of the v2019 communication system stands at 16ms per packet in experimental environment. While this rate is generally adequate for most 8-robot scenarios, there are situations that demand even faster response times, e.g. when a goalkeeper needs to react swiftly to save an attack that breaks through the penalty area, the robots often struggle to respond in time due to the communication delay. Additionally, it's important to note that the data packet communication mode of the 24L01 is limited to 32 bytes per transmission. As a result, we can only accommodate the communication information of up to 4 robots within each packet and endure a certain amount of data compression [3]. Another limitation we have encountered in the v2019 communication system is the high packet loss when there is interference from other 2.4GHz devices, which poses significant challenges to the robot's operation. Since 2.4G type equipment is relatively common in competition venues, we have to choose an area with less interference to place the transmitter, use a larger antenna, or adjust the antenna pointing to mitigate the impact of the problem.

By implementing the new WiFi communication system, the limitations above have been effectively overcome. The communication rate from the host to the robot has significantly improved to around 4.7ms per packet in experimental environment. By utilizing the protobuf-based UDP Communication over WiFi, the limitations we previously encountered regarding data compression or byte control are no longer applicable. Furthermore, during extensive testing with a single robot in the center of the field, operating at a communication rate of 74Hz, we conducted a thorough examination comprising 10,000 consecutive Ping(Packet Internet Groper) tests and did not observe any packet loss throughout the testing process.

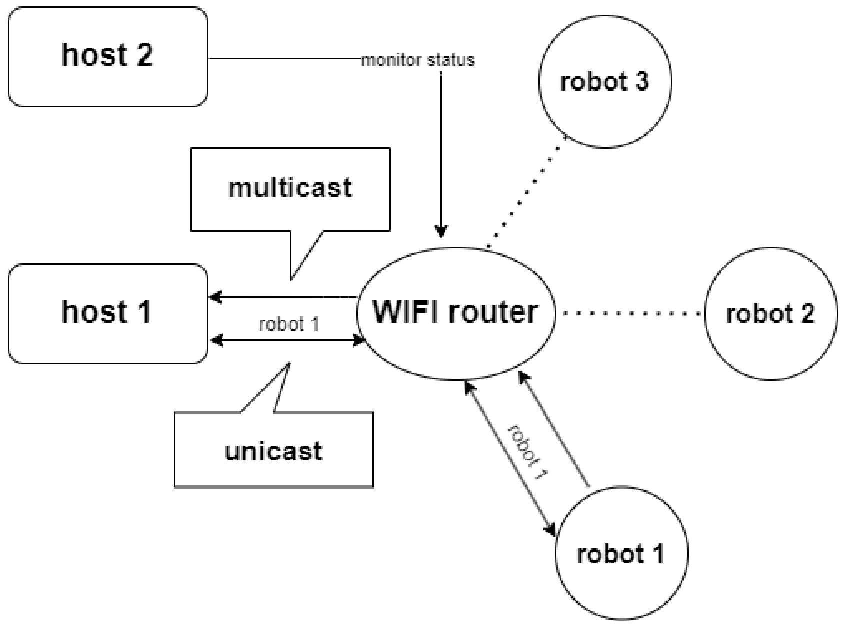


Fig. 2: WiFi communication system diagram. There are two communication paths between host 1 and robot 1, one is unidirectional multicast communication from robot 1 to host 1. The other is a two-way unicast communication between robot 1 and host 1, which is possible once the ip addresses are clear.

In the new communication system, multicast transmission can be implemented through routers. However, to ensure the accuracy of command and report communication, unicast is more advantageous. This brings a challenge: the robot does not initially know the IP address of the command machine, and likewise, the computer host does not know the IP address of the robot. To address this, each online robot performs a basic broadcast within the local area network (LAN) to send out essential information, including its number, vision pattern, battery status, and other relevant details. By doing so, the host can obtain the IP address of the robot, enabling point-to-point (p2p) communication. Once the host has acquired the IP address of a robot, it can establish direct communication with that specific robot. Similarly, when the robot receives packets from the computer, it can extract the host's IP address and include it in the return packet, which can be sent back to the host. Fig 2 shows the basic principle of our dual-path communication and the following pseudo-code lists the main contents of the protobuf.

Protofile 1 Proto for multicast

```
message :
  IP address
  uuid of the robot
  team of the robot
  number of the robot
  battery voltage
  capacitance
  ...
```

Protofile 2 Proto for unicast

```
message :
  uuid of the robot
  infrared signal
  elasped time since last kick
  elasped time since last chip
  imu data
  battery voltage
  capacitance
  wheel encoder value
  team of the robot
  kickmode
  kickpower
  report
  ...
```

To conclude, there are actually two communication methods in the v2023 system, one is the broadcast from the robot, and the other is the p2p communication between the computer and the robot. These two parts are implemented using multicast and single-point communication of the UDP protocol respectively. It should be noted that UDP has three modes: single point, multicast and broadcast; multicast requires the transmitter to notify the router to add the packet to the corresponding group before sending it, while broadcast does not need to notify the router and defaults to all subnet nodes are sent; so when using UDP, it is multicast rather than broadcast that is being utilized.

3 Software

In the past year, as we rejoined the world competition after the pandemic, we summarized all the development since 2020, coming up with a more modular structured decision module which will be introduced in Section 3.1. In the mean time, driven by the demand of promoting the overall offensive efficiency in the increasingly fast-tempo matches and inspired by tactics from the real-world soccer games, we made some adjustment to our attacking and defending strategies to adapt to the 11 vs 11 games, which will be introduced in section 3.2 and 3.3.

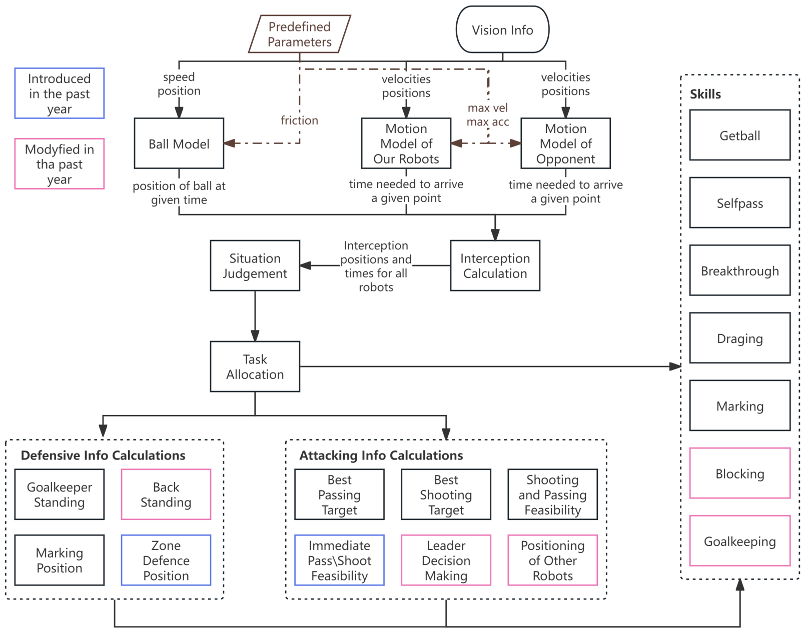


Fig. 3: Overall structure of decision module. Ball model and motion models are used to perform the interception calculation. The situation judgement module will select between attacking and defending based on interception information. Then we'll generate different tasks and call according skills.

3.1 Modular Decision Making

Since the introduction of a new decision module in 2018 [2], we've implemented various new algorithms under the framework to improve the performance of the system from different aspects, like the interception prediction which enhances the performance of skills [3], the dynamic passing module v1 [4] and v2 [1] which greatly improves the passing decisions, etc. Nevertheless, the program became increasingly complex during the process, making it significantly harder to maintain and develop. Therefore, in the past year, we re-organize all the modules we've developed during the years to make the overall decision process more concise. We made great efforts to organize different components in a modular way for easier in-game adjustment and better extensibility.

Fig 3 displays the overall structure of our current decision module. The parts that was introduced or modified in the past year were marked blue and pink respectively. Generally, we use the vision info to calculate the interception information using search-based method based on the models of robot motion and ball movement as introduced in [3]. Then we judge the current situation on the court, e.g. whether we control the ball, based on these information. The judgements are then utilized to generate and allocate different tasks for our robots. In the mean time, we will perform various calculations concerning defending and attacking to generate necessary information for the planning and executing of skills.

Through the freshly-organized modular decision module, implementing new modules or modifying current ones has become easier, as one can easily find the corresponding parts. The overall decision process is also quite straightforward, making it easier to analyse robots' behavior on the court and make proper adjustments accordingly.

3.2 More Efficient Attacking Strategy

As the number of robots expanded to 11, similar to that of a real soccer game, the cooperation between robots has become increasingly important. Meanwhile, with the development of defensive tactics and the evolution of robots' motion capability, effective offence has become more difficult. Inspired by the Tiki-Taka tactics from the real soccer matches, we redesigned our off-ball positioning algorithm as well as the leader decision making, aiming at more fluent attacking with more passes. Moreover, we also add a new module to calculate the feasibility of immediate passing and shooting to promote the overall offensive efficiency.

New positioning for better formation In a 11 vs 11 game, we'll typically arrange 4-6 robots to participate in attacking. Without proper positioning of robots, the front court could become quite crowded, making it hard for both passing and individual skills to execute. Therefore, as part of our approach to replicate Tiki-Taka style of offence, we redesigned the off-ball running module. The main purpose was to provide more passing options for the leader, while also better spacing the front court.

While the overall structure remained the same as introduced in our 2020 ETDP [5], we mainly adjusted two parts of the module. Firstly, we designed a new zone splitting based on the positions of human soccer teams, capable of providing at most 6 off-ball running positions. Secondly, we introduced a dual-scoring evaluation that not only searches for positions with threat to goal but also exploring candidates that could provide passing opportunities to benefit possession-style offence.

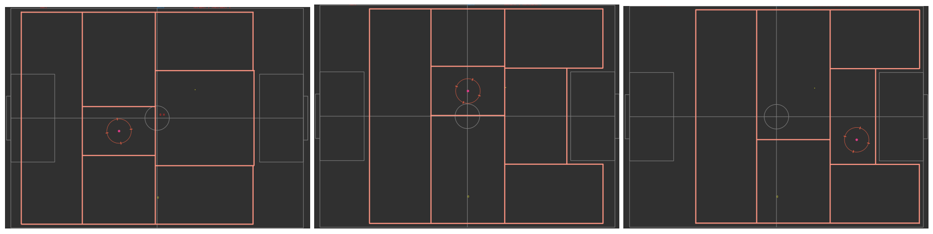


Fig. 4: Off-ball running zone split according to ball position. We'll select one candidate running point from all zones except the one with the ball.

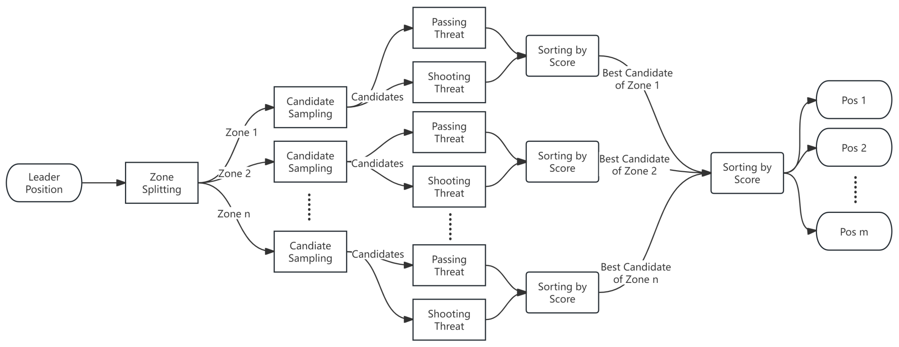


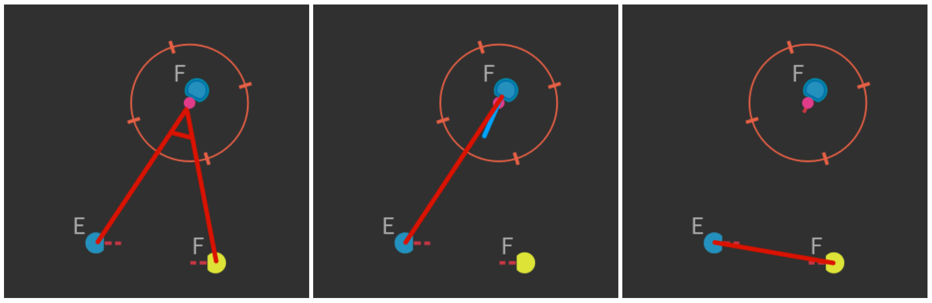
Fig. 5: Workflow of dual-scoring evaluation for off-ball running. We sample candidate points in all zones, calculating the passing and shooting score of candidates. The points with the highest scores will be selected.

As shown on Fig 4, our new splitting method divides the front court to 6 parts, which will be adjusted according to leader's position, providing support from all directions while avoiding the areas around the leader. The zones are similar to the positions of real soccer games, e.g. wings, forwards, midfielders, etc, which could better space the field.

Fig 5 briefly explains the dual-scoring evaluation framework. We implement a new evaluation of passing threat considering passing angle and distance, available spaces and so on. The passing and shooting threat of all the candidates from every zone will be evaluated to generate normalized scores ranging from 0 to 100. Then the position with the highest score will be selected as the candidate of that zone. Finally, we will select the top-scoring zones based on the number of offensive robots to generate the off-ball running positions.

To be more specific, the passing threat is evaluated based on the following criteria: available passing angle, passing turning angle and free spaces around.

- Available passing angle is measured by the smallest angle between the line connecting leader to the candidate and that to an opponent robot, as sketched on Fig 6a. The larger the angle is, the harder it is for the pass to be intercepted, and thus higher the score will be.
- Passing turning angle is assessed with the angle between the leader's orientation and the line connecting leader to the candidate point, as shown on Fig 6b. It's a good representation of how much time the leader would need to actually make the pass. The smaller the angle is, the quicker the leader can make the pass, resulting in a more fluent and dynamic offence.
- Free spaces around is evaluated quite straightforwardly by the distance between the candidate point and the nearest opponent robot. The larger the distance is, the more space the receiver would have, and thus more room for further operations.



(a) Available passing angle. Measured with the angle between two red lines.
(b) Passing turning angle. Measured with the angle between the red and blue lines.
(c) Free spaces around. Measured with the length of the read line.

Fig. 6: Evaluation of passing threat

All the above elements will be normalized with a min-max normalization. Then we'll take a weighted average to produce a score which represents the passing threat. We recommend readers refer to section 6.2 of our 2020 ETDP [5] for the evaluation of shooting threat.

Immediate Passing and Shooting Judgement In our previous leader decision workflow, we'll first judge the feasibility of shooting based on a module to calculate the best shooting target. If shooting isn't available, we'll use the best result from our DPPS [4] and selfpass [1] module as the passing target. Then the algorithm will call appropriate skill to try to get the ball to the chosen target. This approach worked quite well in the past competitions. Yet as the intensity of defense increased greatly over the years, we found that as the leader is always trying to pass or shoot with the optimal target, taking the action will need certain time. The defender would thus have opportunity to get closer to our leader, making it harder for the action to be completed.

Therefore, we implemented a immediate shooting and passing feasibility judgement module. The judgement is based on a simple physical calculation of velocity integration. We'll take the current angular velocity $\boldsymbol{\omega}$, linear velocity $\boldsymbol{v}$ and desired speed of ball $\boldsymbol{v_{des}}$ to calculate the actual ball speed with

$$\boldsymbol{v}_{actual} = \boldsymbol{\omega} \times \boldsymbol{r} + \boldsymbol{v}_{des} + \boldsymbol{v} \tag{1}$$

where $\boldsymbol{r}$ represents the vector pointing from the center of a robot to the center of the ball. The desired speed $\boldsymbol{v}_{des}$ is ball speed limit, i.e. 6.5 m/s, in the case of shooting and the result from DPPS module in the case of passing. Using this equation, we'll be able to predict the actual trajectory and speed of the ball if the robot kicks the ball instantly. Immediate shooting is considered to be feasible if the ball's trajectory goes into the goal, and our interception calculation module indicates that all enemies can't block the shot. As for immediate passing, we'll iterate through all candidate passing points from DPPS. Given that the interception information is already checked within the DPPS module, as long as the candidate is close enough to the trajectory of ball, immediate pass will be considered viable.

After introducing the immediate passing and shooting feasibility judgement module, we'll prioritise direct actions to increase the efficiency of attacking. The overall attacking decision workflow for leader is shown in Fig 7. Instant actions are undoubtedly faster. Yet in the case of passing, an available option may not always be a decent choice. Therefore we added some simple sanity checks for direct passing, e.g. don't pass to backcourt, to prevent undesirable actions.

3.3 Improved Defensive Tactics

Traditionally, as most teams does, we arrange two robots to always stay in front of the penalty area, on the line connecting the ball to the goal center, to block the possible shots of opponent team. This approach has been demonstrated over the years to be very effective. However, as the goal became wider with the introduction of 11 robots, we found that two robots couldn't block the shots effectively. We tried to increase the number of blockers to four, which did better in most cases. Yet this approach still struggled with rapid passes across the penalty area. Therefore, we designed a brand new defense system, including a

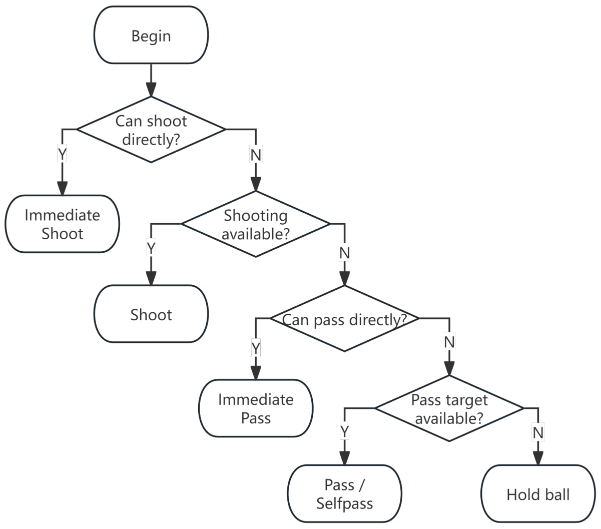
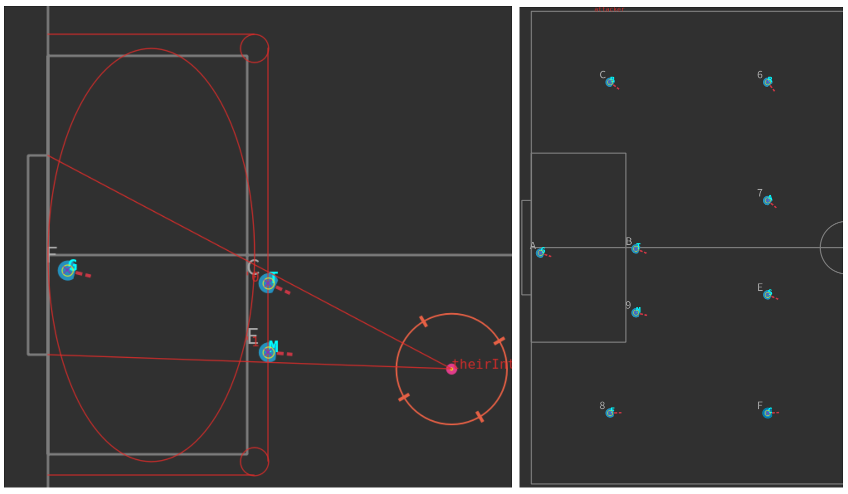


Fig. 7: Decision tree for leader with immediate passing and shooting



(a) New blocking positions (b) Zone defence positions

Fig. 8: New defense system

new blocking approach, a new zone defense tactic and cooperation strategies between defensive roles.

New blocking approach As we reconsidered the original blocking approach, we realized that the blockers were actually doing the same job as the goalkeeper. Inspired by the cooperative strategies between backs and keeper in real soccer games, we came up with a new blocking algorithm. As shown on Fig 8a, the blockers' position is determined by the lines connecting ball to two posts. With the new arrangement, goalkeeper and blockers are responsible for shots to the middle and to the sides respectively, which indeed narrows the angle that the goalkeeper needs to cover.

Zone defense system Conventionally, apart from the goalkeeper and the blockers, we arrange a marking robot for each opponent robot on our backcourt, while other robots remain on the frontcourt preparing for counter attack. The issue with this approach is that when an opponent moves from our frontcourt to our backcourt, we'll have to dispatch a robot from the frontcourt to play the marking role. This could result in disorders in marking priorities, i.e. the correspondence between our marking robot and opponent attacker, leading to chances for opponents.

Therefore, we utilize the tactic of zone defence, arranging robots to stay in the points as shown on Fig 8b, preparing for marking or other defensive tasks. Whenever a new opponent is present on our backcourt, we'll dispatch the robot from the nearest zone to perform the marking task. Also, the zone defence approach could also limit opponents' passing choices.

4 Standard Benchmark

As we designed v2023 robot in the past year, we found that the motion, dribbling and kicking capabilities were quite different from that of the v2019 robot. Therefore, we introduced a series of benchmark to examine the robot's capabilities as well as the control algorithms from various perspective in a objective way.

4.1 Benchmark Items

TestMove This test mainly targets at robots' motion abilities, including acceleration as well as top speed. The test method is rather simple: let the robot accelerate as fast as possible to its top speed with a fixed orientation. As the robot is not central symmetric, we test on both the X and Y axis, as shown on Fig 9. If the robot's direction error, i.e. the angle between its current direction and its initial direction, is larger than $\pi/10$, the trial run is considered fail. We record the highest acceleration as well as the top speed to represent the motion capabilities.

TestRectangle This test also examines robots' motion capabilities, focusing on their performances to tracking straight lines and making sharp turns. The robot should try to track a rectangle of $6000 * 4000$ clockwise, as shown on Fig 10a. We record the robot's position and the vertical distance between the

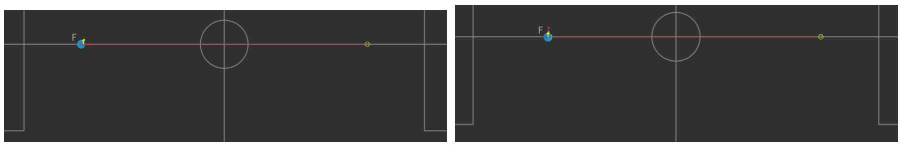
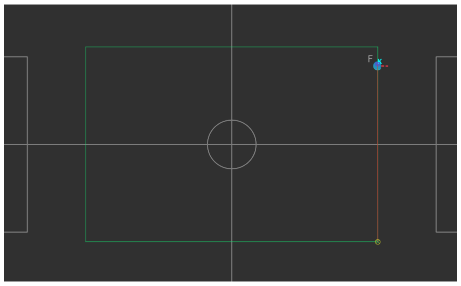


Fig. 9: TestMove running on X and Y axis of robot

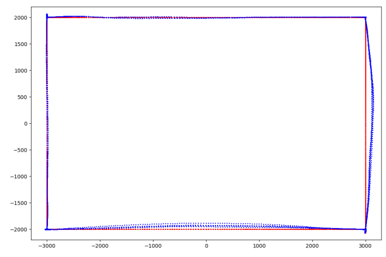
robot and the rectangle, denoted as d_t, on every frame of vision info received. We measure the overall tracking error with a ratio defined by

$$err = \frac{\sum_t d_t}{L} \tag{2}$$

where L represents the total length of trajectory. Typically, we'll let the robot run 5 cycles to eliminate impact from random disturbance. Fig 10b shows an example of robot positions during the test.



(a) Robot running TestRectangle



(b) Example result

Fig. 10: Demonstration of TestRectangle

TestKick This test is designed to test the robots' kicking abilities. To test stability, robots should try to kick the ball at a speed of $6.0m/s$ for 20 times and chip the ball to a distance of $4.0m$ for 20 times. The variance of results will be used to access the stability of relative mechanical structures. Also, we use the maximum speed of flat kick and maximum distance of chip kick to represent the potential capabilities of kicking.

TestDribble Dribbling is also a critical capability of SSL robots. We use three indexes to represent dribbling capabilities from different aspects. For dribbling with translation, we let the robot dribble the ball in stationary and accelerate with $a = 3.0m/s^2$ on directions of Y and $-X$ axis. And for dribbling with rotation, we let robot dribble the ball and slowly increase its rotation speed. The top linear or angular speed before the robot lose control of the ball is recorded to represent the dribbling capability.

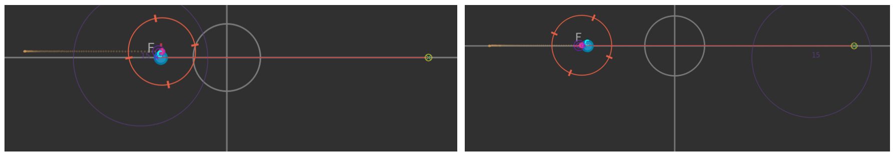


Fig. 11: TestDribble on Y and $-X$

4.2 Results

We use the above benchmark to test our v2023 and v2019 robots, the results are listed in the following table.

Item		**v2023**	**v2019**
TestMove	$v_x(mm/s)$	4000	4000
	$a_x(mm/s^2)$	6000	4455
	$v_y(mm/s)$	4000	4000
	$a_y(mm/s^2)$	5000	3606
TestRectangle	err	0.03	0.03
TestKick	max $v_{flat}(m/s)$	7.1	6.6
	$var_{flat}(m^2/s^2)$	0.033	0.023
	max $d_{chip}(m)$	5.0	6.4
	$var_{chip}(m^2)$	0.015	0.028
TestDribble	$v_x(m/s)$	2.3	3.9
	$v_y(m/s)$	2.6	3.9
	$v_w(rad/s)$	9.3	12.9

Table 1: Benchmark result of v2023 and v2019 robot

It's clear that our v2023 robot possesses a significantly advantage in terms of motion capabilities compared to v2019 robot. Yet the dirbbling ability is slightly weaker.

We sincerely welcome all the SSL teams to benchmark their robots based on the benchmark items. Hopefully our benchmark standard could provide a fair and objective way to compare robots within the SSL community, encouraging teams to pushing the robots' capabilities to a higher level.

References

1. Huang Z, Han C, Shen N, et al. ZJUNlict Extended Team Description Paper[J].

2. Chen L, Jin L, Lon C W, et al. ZJUNlict extended team description paper for RoboCup 2018[J]. RoboCup Wiki as extended team description of ZJUNlict, Montreal, Canada, accessed March, 2018, 6: 2019.
3. Huang Z, Chen L, Li J, et al. ZJUNlict extended team description paper for RoboCup 2019[J]. arXiv preprint arXiv:1905.09157, 2019.
4. Chen Z, Zhang H, Guo D, et al. Champion team paper: Dynamic passing-shooting algorithm of the RoboCup soccer SSL 2019 champion[C]//RoboCup 2019: Robot World Cup XXIII 23. Springer International Publishing, 2019: 479-490.
5. Huang Z, Zhang H, Guo D, et al. Zjunlict extended team description paper for robocup 2020[J].