# ZJUNlict Team Description Paper

**Humanoid Kid-Size League of Robocup 2023**

Zheyuan Huang, Jiangpin Liu, Jialei Yang, Jiazheng Yu, Jiaxi Huang, Wei Yu, Chunlin Zhou, 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.** In this paper, we show our latest progress, including mechatronics system of our new robot and the improvement of the software part, including self-localization and decision part. For self-localization part, we tested several existing visual inertial odometry(VIO) algorithms and integrated them into our positioning framework. In the decision-making section, we focused on making changes to goalkeeper decisions. Finally, we open source a streamlined multi-machine communication framework that aims to replace ROS[1].

**Keywords:** Humanoid Robot · self-localization · decision-making · ROS

## 1 Introduction

The ZJUNlict 2023 team from Zhejiang University (China) builds upon the ZJUDancer team, which consists of mainly undergraduate students. It's been three years since our ZJUDancer team last competed in RoboCup 2019 so our team members participating this year are all rookies.

This paper presents the details of our works since the Team Description Paper of 2019 [1], including a new circuit system design and some improvements in self-positioning, decision-making of goalkeeper and communicating in multi-robot. In the next section, we present an overview of our newly designed robot. In section 5, we summarizes our contributions and discusses future work.

## 2 Overview

In 2019, we developed a robot based on Nvidia TX2 and achieved the second place. Due to the excellent GPU performance of TX2, we can perform object detection based on neural network. However, in the subsequent development, we realized that the CPU resources of TX2 could no longer meet our needs. Another limitation of our previous version of the robot is mainly in the communication

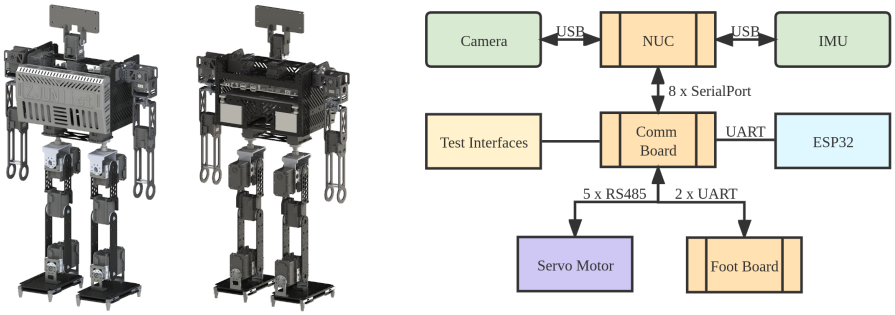
[1] https://github.com/ZJUNlict/zos

Table 1: Robot Specifications.

| **Robot version** | **v2019** | **v2023** |
|---|---|---|
| CPU Board | Nvidia Jetson TX2 | Intel NUC 11 Enthusiast |
| Carrier Board | Orbitty ASG003 | None |
| OS | Ubuntu 16.04 | Ubuntu 22.04 |
| Interfaces | 1 x Ethernet<br>2 x USB 2.0 | 1 x Ethernet<br>6 x USB 3.0 Type-A<br>2 x USB 3.0 Type-C |
| Communication Chip | STM32+FT232RL+MAX3443 | CH348L+MAX3485 |
| Number of DOF | 18 | 20 |
| Height | 670mm | 680mm |
| Width | 260mm | 370mm |
| Weight | ≈ 4kg | ≈ 5kg |
| IMU | ADIS16355 | XSENS MTi-300-2A8G4 |
| Camera | OmniVision OV2710 | MYNT-EYE-S2110 |
| Motor | DYNAMIXEL MX28, MX64, MX106 | |
| Walking Speed | ≈ 20cm/s | |
| Battery | 4s(14.8v, 2000mAh) | 6s(22.2v, 5300mAh) |

with the steering gear and the soles of the feet. Due to delays, packet loss and other reasons, the upper computer cannot obtain feedback data in time. In order to solve the above two problems, we designed a new robot. The comparison of the two robots is shown in Table 1.

Our newly developed robot LEI in Figure 1-a uses Intel NUC to replace the original TX2 as the main controller. The comparison of its computing resources is shown in Table 2. Compared with TX2, the new main controller has sufficient computing resources so that we can implement more complex algorithms and more timely IO responses. Compared with the old version that uses the STM32 chip to communicate with the main controller through a single serial port, the new version can put more IO pressure on the host computer to improve the real-time performance of sensor feedback. The overall communicating architecture of our robot is shown in Figure 1-b, and then in section 3.1 we show the new version of the foot board and in section 3.2 we introduce the specific implementation of the communication board.




(a) The front and back renderings (b) Communication Architecture

Fig. 1: Hardware Design

Table 2: Main Controller Specifications.

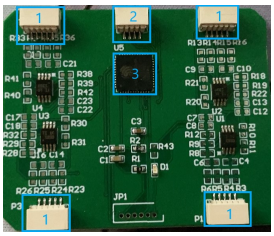
| **version** | **v2019** | **v2023** |
|---|---|---|
| Name | Nvidia Jetson TX2 | Intel NUC 11 Enthusiast |
| CPU | Dual-Core NVIDIA Denver 2 Quad-Core ARM Cortex-A57 | Intel i7-1165G7 |
| Total Cores | 2+4 | 4 |
| Total Threads | 6 | 8 |
| Geekbench Multi-Core Score | Lower than 2834(Nvidia NX) | 4411 |
| GPU | 256-core NVIDIA Pascal | Nvidia Geforce RTX 2060 |
| CUDA cores | 256 | 1920 |

# 3 Hardware

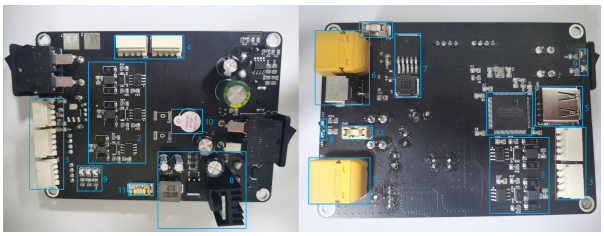
## 3.1 Communicating & Power Board

The robot's limbs and head need a total of five channels of RS485 communication. With two 1000Hz UART foot boards and an ESP32 for debugging, a total of eight channels of serial communication are required. Therefore, we use the CH348L chip, which supports 6Mbps independent eight channels of serial communication. The front and back of the communication board are as shown in Figure 2-b. This circuit can also power the servo motor and the foot board, accepting a wide voltage input of 24-40V. There are two switches that separately control the power supply of the servo motor, the foot board module, and the NUC. The power supply mode can support two kinds of battery and external power supply, and in the case of external power supply, it will actively disconnect the battery power supply. We use the LM2576 chip for 3.3v voltage regulation, and the XL4016 chip for 14.8v voltage regulation, which are separately used for the

power supply of the foot board and the servo motor. The board also has indicator light and buzzer for debugging and low voltage alarm.



(a) ① Connector to strain gauges. ② UART Connector to main PC. ③ ESP32



(b) ① CH348L for 8 serial ports. ② 5 MAX3485 for RS485 to serial conversion. ③ RS485 connector to servo motors. ④ UART connector to foot board. ⑤ serial to NUC. ⑥ connector for battery and external power. ⑦ 3.3v regulator circuit. ⑧ 14.8v regulator circuit. ⑨ LEDs for debugging. ⑩ buzzer for low voltage alarm. ⑪ fuses

Fig. 2: Foot Board & Communication Board

### 3.2 Foot Board

This year we redesigned a foot board (as shown in Figure 2-a), using a 2x2 strain gauge matrix. Filtering and signal amplification are required when collecting strain gauge signals and we adopt the INA333 chip to ensure good accuracy. On the main chip, we use ESP32 instead of STM32, and use UART to directly communicate with the main NUC board. Compared with the previous version, this scheme has the following advantages

- Compared with the RS485 communication of the previous version, the new version of the communication can achieve higher frequency feedback independently of the servo communication.
- The AP function carried by ESP32 enables the status of the sole board to be tested directly through WIFI communication, which is convenient for hardware debugging.

## 4 Software

### 4.1 VIO Algorithm Test

In recent years, research on VIO algorithms such as VINS and ORB-SLAM has developed rapidly. Combining VIO to complete self-positioning can effectively improve positioning accuracy. For the humanoid field, we have tested the three algorithms of VINS-Fusion[2], OpenVINS[3] and ORB-SLAM3[4] using AMD

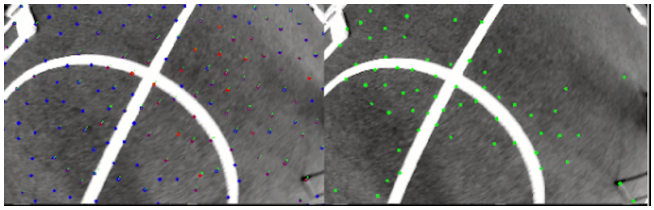
Table 3: VIO Algorithms Compare

| Algorithms | CPU Usage | Mem Usage | Error of Loop ( % ) |
|---|---|---|---|
| **VINS-Fusion** | 270 | 2.4 | 2.6 |
| **OpenVINS** | 100.7 | 2.1 | 4.3 |
| **ORB-SLAM3** | 265 | 13.2 | 2.2 |

2700x and 16G 3200MHz memory and test results are shown in the table 3. Visualizations of some test data are shown in Figure3.

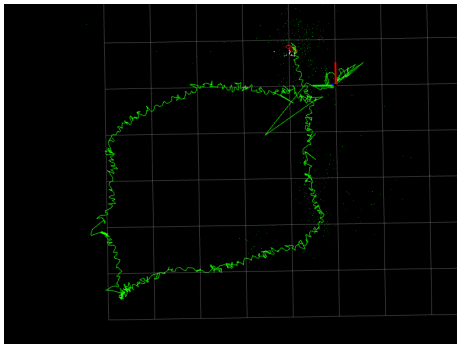
During the test, we encountered the following problems.

- When testing on a robot, the camera shakes violently. So it is necessary to use the gimbal for the camera.
- The VIO algorithm is completely unable to cover the situation where the robot falls, so special handling is required for this situation.
- The robustness of ORB-SLAM3 is not as good as the other two algorithms. We guess that there may be more stringent requirements for the calibration of the IMU. Further testing is required.

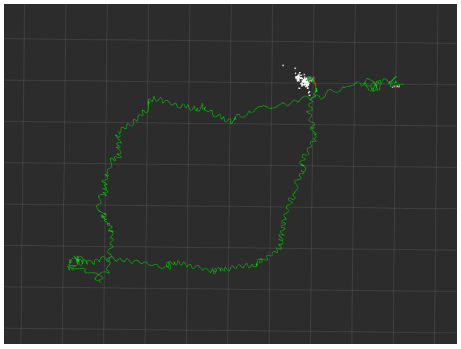


(a) VINS-Fusion Features

(b) ORB-SLAM Features



(c) VINS-Fusion on Robot



(d) OpenVINS on Robot

Fig. 3: VIO Test

### 4.2 Decision of Goalkeeper

Our behavior module is still based on a mixture of behavior tree and finite state machine, implemented in python. To improve the defend strategy, we add a new role of goalkeeper to the behavior module this year. The basic rules of this role are as follows. There are three main parts of the strategy due to the three statuses of the ball:

- When the ball is not detected by the robot, the robot stays in the middle of the gate and scan around to get the visible information of the ball.
- When the ball is detected and its velocity is over the threshold, it is regarded as a dangerous ball. According to the prediction of the position of the ball , the robot executes the action of save ball and blocks out the ball.
- When the ball is detected with a small velocity, the robot chooses the strategy of attack. When the ball is in margin position regarding to goal area, then attack.

### 4.3 Alternative framework for ROS

We started using ROS[5] as middleware in 2017 to help us handle inter-process communication. To be honest, ROS is really a great framework, its robust communication and easy-to-use API greatly shorten our development time. This year, we tried to implement a set of middleware for communication by ourselves for the following reasons. First, we want to convert the communication encoding to the widely used Google Protobuf. Secondly, in the case of multi-machine communication, ROS requires more complex configurations to achieve communication, and problems are prone to occur when the only ros-master connection is unstable. Finally, we hope that UDP with lower latency can be used instead of TCP for communication. So we refer to the Data Distribution Service used by ROS2 and realize ZOS. The basic framework of zos for communication is shown in Figure 4.

A complete message communication process has the following 4 steps.

- Each process sends its own subscriber or publisher information to the multicast channel, and at the same time receives the topic list sent by all other processes under the current LAN from the multicast channel.
- When a subscriber exists in a process and an external process has a publisher to send a message with the same name, the subscriber of the current process will bind a udp port and send a Remote Procedure Call (RPC) request to the target process.
- After the target process receives the RPC request, it will add the udp address in the request parameter to the publisher in the process.
- The next time the publisher has a message to send, it will traverse the entire subscription list to send data. The target process responds to the RPC request according to the current publisher situation to complete the entire message discovery process.

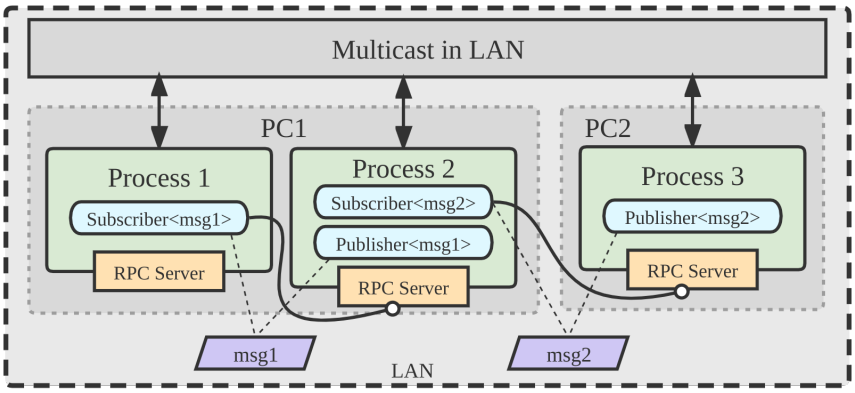



Fig. 4: ZOS Architecture

## 5 Future Works

There are still many deficiencies in the work we have shown so far. In the remaining time we will focus on the following points.

- In the previous work, we have used the YOLO framework for object detection. In the follow-up work, we will improve robot recognition and friend-or-foe recognition to achieve stronger perception capabilities.
- In the past, the strategy planning of robots was relatively simple for the situation of the court. We will consider adding more targeted football tactics such as pressing and marking after the perception ability is improved.
- For the force of high-frequency feedback, we will carry out real-time gait planning based on MPC combined with ZMP criterion.
- There are a large number of parameters in the robot's gait, and it is difficult to debug without using good visualization tools. A visualization tool for gait parameters will be developed later.

## 6 Conclusion

This paper describes some of the progress we have made on this new robotic system, but there are still many improvements that can be made. The new robotic system has not yet passed complete testing, and the self-localization framework combined with the VIO algorithm has not yet been completed. We have not participated in international competitions for three years. Most of the participants this year are undergraduates and they are participating for the first time. We hope to make more improvements in the next six months. Looking forward to seeing you all.

## References

1. Fan W, Chen X, Jiang J, et al. ZJUDancer team description paper[J]. Technical report, State Key Laboratory of Industrial Control Technology, 2019.
2. Qin T, Pan J, Cao S, et al. A general optimization-based framework for local odometry estimation with multiple sensors[J]. arXiv preprint arXiv:1901.03638, 2019.
3. Geneva P, Eckenhoff K, Lee W, et al. Openvins: A research platform for visual-inertial estimation[C]//2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020: 4666-4672.
4. Campos C, Elvira R, Rodríguez J J G, et al. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam[J]. IEEE Transactions on Robotics, 2021, 37(6): 1874-1890.
5. Quigley M, Conley K, Gerkey B, et al. ROS: an open-source Robot Operating System[C]//ICRA workshop on open source software. 2009, 3(3.2): 5.