NBT Team Description Paper

Kehan Li[1], Jincheng Ni[1], and Yifan Cai[1]

School of Information Science and Engineering
NingboTech University
No. 1 Qianhu South Road, Yinzhou District, Ningbo,
Zhejiang Province, P.R. China
`ausksyc@outlook.com`
https://www.nbt.edu.cn/

Abstract. This paper presents the NBT team's work over the past year, a Small Size League (SSL) team intending to participate in RoboCup 2026 in Korea. The paper details the team's hardware development, software system design, and key optimization using CUDA-based parallel computing to meet the real-time requirements of RoboCup SSL competitions.

Keywords: RoboCup SSL · Soccer Robot · CUDA · Parallel Computing · Real-Time Control

1 Introduction

The NBT team is composed of students from NingboTech University. Since 2018, we have been engaged in small-scale soccer robot-related research and competitions, and have actively participated in various domestic robotics events in China. Through continuous practice and exploration in these domestic competitions, we have gradually accumulated basic experience in robot development, system debugging and competition coordination.

This year, we are honored to have the opportunity to participate in the RoboCup Small Size League (SSL) as our first international competition. We aim to take this opportunity to learn from excellent teams around the world, exchange technical experience, and broaden our horizons.

Aware of the gaps between our team and international advanced levels, we have made ample preparations for our self-designed small-scale soccer robots. We will strive to meet the relevant requirements of the RoboCup SSL, and wholeheartedly participate in this international event with a learning attitude.

2 Hardware

Our team has developed two generations of small-scale soccer robots so far. The R&D process of both generations has received strong support and assistance from many domestic teams (such as ZJUNlict and SRC), companies (including Nanjiang ILOBOKE and TurningZero), and our university. Their guidance and help have laid a solid foundation for the iterative upgrade of our robots.

2.1 First-Generation Robots

In 2022, we meticulously designed a batch of first-generation robots, with the STM32F407ZET6 microcontroller as the core control unit. We clearly recognize that these first-generation robots still have significant room for improvement in terms of performance and stability. Consequently, we continuously seek experienced professionals in mechanical design, circuit design, and embedded development to join our team. It is heartening to note that several highly proficient technicians capable of replicating and further enhancing our designs have already come on board. With their efforts, we have steadily progressed towards the goal of refining our robot designs, laying the groundwork for the development of the second-generation robots and paving the way for our participation in international competitions such as RoboCup.

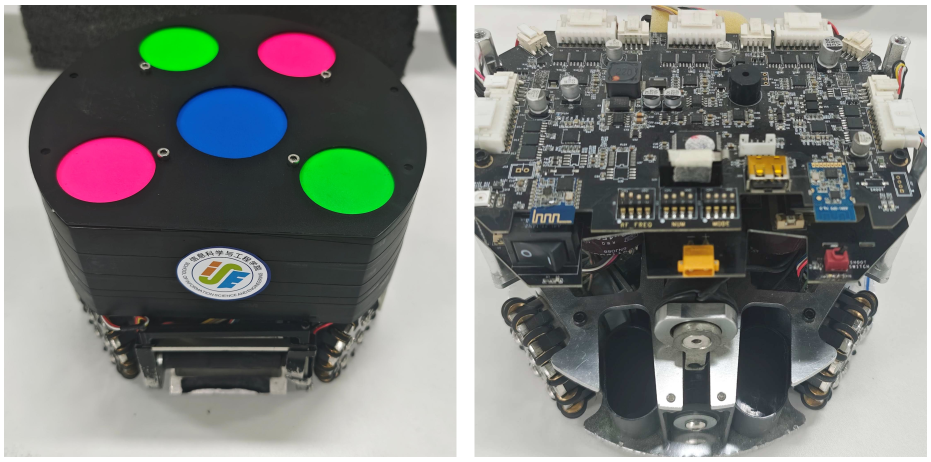
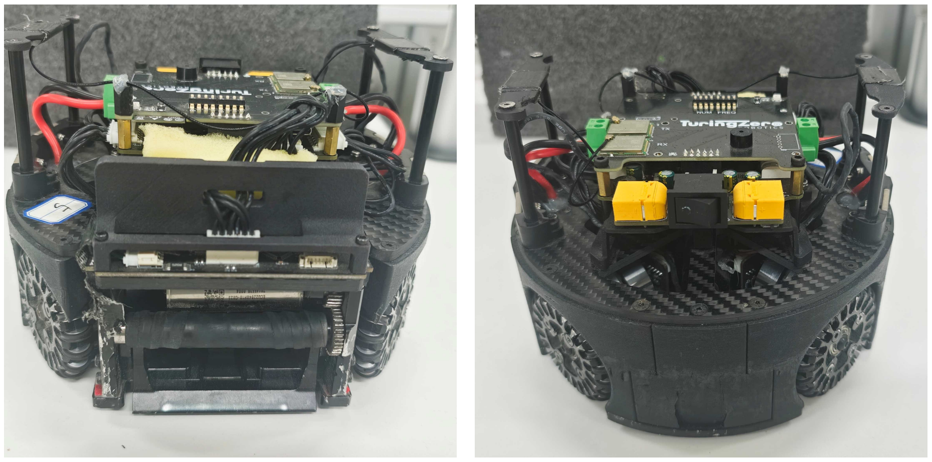

(a) External appearance of Robot V1 (b) Internal structure of Robot V1

Fig. 1: Overall Structure of SSL Robot V1

2.2 Second-Generation Robots

Building on the foundation of the first-generation robots, we have upgraded and optimized key components to develop the second-generation robots, aiming to improve their competitive performance. Specifically, we replaced the original ball-suction motor and omni-directional wheel motor with higher-performance alternatives, which enhances the robot's ball-handling capability and movement flexibility. Meanwhile, drawing on the design experience of various international teams, we replaced the suction nozzle material with polyurethane to improve the stability and durability of ball adsorption. In terms of the control unit, the microcontroller was upgraded from STM32F407ZET6 to ESP8266, which offers better wireless communication support and lower power consumption. It should be noted that both the first and second generations of robots adopt NRF24L01 as the communication module to ensure stable data transmission during competitions.



(a) Front view of Robot V2 (b) Rear view of Robot V2

Fig. 2: Overall Structure of SSL Robot V2

2.3 Communication System

The communication system of the NBT team's SSL robots is designed to realize reliable bidirectional data transmission between the host computer and on-field robots, supporting collaborative operation of two generations of robots while meeting the real-time and stability requirements of RoboCup competitions. Built on the Qt C++ framework, the system adopts a master-slave architecture and integrates hardware modules, protocol specifications, and software logic to form a closed-loop communication pipeline.

Hardware Foundation The communication hardware is centered on wireless transceivers and microcontrollers, with consistent core components across two generations of robots to ensure compatibility and scalability:

– **Wireless Communication Module**: Both generations of robots utilize the NRF24L01 2.4GHz wireless transceiver module as the core communication unit. Operating in ShockBurst™ mode, this module enables high-speed data transmission with low power consumption, establishing a stable bidirectional data link between the host computer and robots.

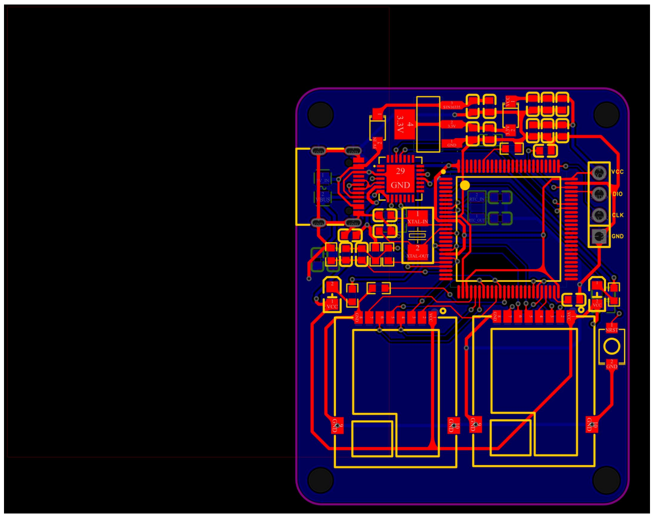


Fig. 3: NRF24L01 Transmitter PCB Drawing (for Host Computer)

- **Robot Control Unit**: The first-generation robots adopt the STM32F407ZET6 microcontroller, while the second-generation robots feature an upgraded microcontroller configuration optimized for better integration with the NRF24L01 module, adapting to the long-duration operation requirements of competitions without changing the core communication protocol or module.
- **Host Computer Interfaces**: The host computer supports serial (RS232/USB) interfaces with dynamic adjustment capabilities to adapt to different competition environments and robot versions, maintaining consistency in core communication logic for both generations of robots.

Communication Protocol Specifications To ensure compatible and reliable data transmission, a customized communication protocol is developed based on the characteristics of the NRF24L01 module, with clear specifications for communication modes, data frame structures, and verification mechanisms:

Communication Modes A unified communication mode based on the NRF24L01 module is implemented for both generations of robots, with adaptive parameter adjustments to match the hardware differences of different robot versions:

- **Serial-adapted Communication for First-generation Robots**: Optimized for the STM32F407ZET6 controller of first-generation robots, it operates at a baud rate of 115200 (8 data bits, no parity bit, 1 stop bit) and follows instruction encoding rules compatible with the hardware architecture of early robots.

- **Optimized Communication for Second-generation Robots**: Adjusted for the upgraded microcontroller of second-generation robots, it retains the core logic of the NRF24L01-based communication protocol and only modifies hardware-adaptive parameters, avoiding cross-team interference through dedicated channel configuration rather than protocol changes.

Data Frame Structure The protocol defines fixed-length data frames for both command transmission and status feedback, ensuring efficient parsing and low communication overhead while maintaining consistency across robot generations:

- **Command Frame (Host Computer → Robot)**: A fixed length of 25 bytes with a unique header identifier for protocol identification. Core fields include robot ID, motion parameters (linear velocities vx/vy, angular velocity vr), kicking power, dribble speed, and kicking mode (flat shot or chip shot). Batch transmission of multiple robot commands is supported to reduce communication frequency and overhead, consistent with the transmission characteristics of the NRF24L01 module.
- **Status Feedback Frame (Robot → Host Computer)**: A fixed length of 26 bytes with a double-byte header for protocol identification. Key feedback fields include robot ID, infrared detection status, kicking execution status, battery voltage, capacitance, and wheel speed, ensuring unified parsing logic for both generations of robots.

Verification and Compatibility Mechanisms

- **Data Integrity Verification**: The CRC8 checksum is adopted for initialization frames. The host computer calculates the checksum and appends it to the end of the frame, ensuring the integrity of configuration data during NRF24L01 module initialization for all robots.
- **Cross-Generation Protocol Compatibility**: To support simultaneous operation of two generations of robots, a protocol distinction mechanism is implemented based on hardware-adaptive parameters rather than protocol upgrades. At startup, the host computer initializes the communication parameters for each robot by reading configuration files to match the hardware characteristics of first/second-generation robots. During data parsing, a specific byte in the feedback frame is checked to distinguish robot generations (a value of 0 for second-generation robots and non-zero for first-generation robots), eliminating data confusion and enabling seamless compatibility without changing the core NRF24L01 communication protocol.

Communication Workflow The communication process follows a standardized closed-loop workflow, ensuring real-time and reliable data interaction based on the NRF24L01 module's operating characteristics:

1. **Initialization and Frequency Configuration**: The host computer transmits a 6-byte initialization frame to configure the communication frequency and operating mode of the NRF24L01 module. Frequency encoding follows specific rules (frequencies < 8 multiplied by 4; frequencies 8 calculated as (frequency $\times 4$ $+58$)) to avoid channel interference, with consistent configuration logic for both robot generations.
2. **Command Encoding and Transmission**: After generating motion control instructions (e.g., speed, kicking commands) via the decision-making module, the host computer encodes the instructions according to the NRF24L01 protocol specifications, with parameter adjustments based on robot IDs to compensate for individual differences between robots. Encoded commands are then transmitted to robots via the NRF24L01 module.
3. **Status Feedback and Parsing**: The host computer monitors the receive buffer in real time to capture feedback frames from robots. Key status data (e.g., infrared detection results, battery level) are parsed and updated to a global data cache for invocation by the decision-making module. For robots with no status feedback within a 1000 ms timeout window, the system automatically resets their status flags to avoid invalid data affecting decision-making.

Robustness Design To address the complex electromagnetic environment of competitions, multiple robustness enhancement measures are integrated into the communication system, all based on optimizing the use of the NRF24L01 module:

– **Batch Transmission**: Packaging multiple robot commands into a single frame reduces communication overhead and improves transmission efficiency, adapting to the high-frequency instruction requirements of multi-robot coordination.
– **Parameter Calibration**: Kicking power parameters are calibrated using a quadratic function (limited between 10 and 127) to ensure consistent kicking performance across different robots, independent of protocol changes.
– **Timeout Handling**: Automatic status reset for robots with no feedback ensures the stability of the decision-making system, preventing malfunctions caused by lost communication links.
– **Cross-Generation Compatibility**: The parameter-based distinction mechanism enables mixed use of two generations of robots, enhancing the system's flexibility and adaptability to different competition scenarios without modifying the core NRF24L01 communication protocol.

Future Optimization Directions While the NRF24L01 module offers advantages of low power consumption and fast transmission, it faces significant electromagnetic interference in competitions due to the widespread use of 2.4GHz frequency band modules by other teams, leading to increased packet

loss and communication delay. To address this issue, the team plans to replace the communication module with SX1280 in future iterations.
The SX1280 module also supports the 2.4GHz frequency band but adopts LoRa modulation technology, which provides stronger anti-interference capability and longer communication distance compared to NRF24L01. Additionally, it supports adaptive data rate adjustment and channel hopping, enabling dynamic avoidance of interfering channels to further improve communication stability. Comparative tests on the communication performance of NRF24L01 and SX1280 will be conducted to verify the feasibility of the replacement scheme, laying the foundation for more reliable communication in international competitions.

2.4 Software Framework Architecture

The software system of the NBT team's SSL robots is built on the open-source **ROCOS framework** developed by the ZJUNlict team, a modular, scalable, and real-time architecture tailored for small-size soccer robot competition systems. The framework decouples the perception, decision-making, and motion control modules, facilitating rapid iterative development and cross-platform deployment for SSL teams.
The core of the ROCOS framework consists of two essential components: the **Client module** and the **Medusa module**, which collaborate via a high-speed data communication protocol to realize the closed-loop control of the "perception-decision-execution" pipeline during competitions. Their specific functions and data interaction logic are elaborated as follows:

- **Client Module**: As the front-end data hub of the system, the Client module undertakes three core data processing tasks. First, it receives raw visual positioning data from the SSL vision system and performs denoising and Kalman filtering to obtain high-precision pose information of robots and the ball. Second, it parses real-time command signals from the game referee box, including match phases (kickoff, penalty kick, etc.), score updates, and foul warnings, to guide the decision-making module in adaptive strategy adjustments. Third, it collects feedback data from on-field robots via the NRF24L01 communication module, such as battery levels, motor working status, and suction cup activation status, to monitor hardware health in real time. After integrating the above multi-source data, the Client module packages and transmits them to the Medusa module through a shared memory mechanism with millisecond-level latency.
- **Medusa Module**: Serving as the core decision-making engine of the system, the Medusa module is responsible for global strategy planning, local path planning, and action command generation. Based on the fused data from the Client module, it first constructs a real-time field map through environment modeling, which includes the positions of teammates, opponents, and the ball, as well as field boundaries and penalty area limits. Then, it executes predefined strategy algorithms to determine the role

assignment of each robot (attacker, defender, goalkeeper) and generates specific action instructions for individual robots: the speed of each omnidirectional wheel to achieve the desired movement trajectory, the activation signal of the suction cup for ball handling, and the selection of shooting modes (flat shot or chip shot). After completing decision calculations, the Medusa module sends the action instructions back to the Client module.

Finally, the Client module converts the received action instructions into serial port data frames and transmits them to on-field robots via the NRF24L01 transmitter. The robots execute corresponding actions according to the instructions, forming a complete closed-loop control flow of the software system.

3 CUDA-Accelerated Parallel Computing Module for Optimal Offensive/Defensive Positioning

3.1 Module Background and Objectives

In the RoboCup Small Size League (SSL), the calculation of optimal passing/shooting positions requires traversing a large number of candidate points across a 3D space of *players-angles-distances*, with each candidate point undergoing kinematic prediction, field legality verification, and safety assessment. Traditional CPU-based serial computing fails to meet the real-time requirement of the competition (single-frame decision latency < 10 ms) due to the high computational overhead of candidate point traversal and multidimensional evaluation.
To address this challenge, a GPU parallel computing module was developed based on the NVIDIA CUDA architecture, with three core objectives:

1. Leverage the many-core parallelism of GPUs to distribute candidate point calculation tasks and reduce single-frame latency for position evaluation;
2. Achieve parameter adaptation for simulation/real-world scenarios to ensure module universality across different test environments;
3. Optimize CPU-GPU interaction logic to minimize data transmission overhead and improve heterogeneous computing efficiency.

3.2 CUDA Parallel Architecture Design

The module adopts a heterogeneous architecture of *CPU control layer + GPU computation layer*, with core designs including thread/block dimension mapping, data interaction mechanisms, and kernel function encapsulation.

Thread/Block Dimension Mapping for Spatial Partitioning For the 3D candidate point space of *players-angles-distances*, a hierarchical CUDA parallel indexing strategy is designed to decompose the computation into fine-grained parallel units:

- **Grid Dimension (Number of Blocks)**: The number of blocks is set to the total number of our players, where each block corresponds to one receiving player. This achieves coarse-grained parallelism at the "player level", ensuring independent computation of candidate points for different players.
- **Block Dimension (Number of Threads)**: A 2D thread layout is adopted, where one thread dimension maps to the angle index of candidate points (covering the full 0–2PI range) and the other maps to the distance index (expanding outward from the player's position at a fixed step size). This achieves fine-grained parallelism for "angle-distance under a single player".
- **Global Index Mapping**: A linear indexing formula is used to map 3D candidate points to a 1D result array, ensuring each thread is responsible for computing all parameters of exactly one candidate point and avoiding data competition between threads.

CPU-GPU Data Interaction Mechanism The core of CPU-GPU interaction lies in solving the problems of *data transmission* and *execution synchronization.* The module adopts the CUDA Unified Memory mechanism for efficient interaction, with the following principles:

1. **Memory Allocation Phase**: The CPU allocates shared memory space via the CUDA Unified Memory interface, which is mapped to both the CPU's virtual address space and the GPU's virtual address space. This allows direct access by both the CPU and GPU without explicit data copying.
2. **Data Input Phase**: The CPU writes input data (player states, ball position, scenario identifiers) to Unified Memory, which is directly readable by the GPU—eliminating the overhead of explicit "host-to-device" data copying in traditional CUDA workflows.
3. **Computation Execution Phase**: After triggering GPU kernel execution, the CPU waits for computation completion via synchronization interfaces to ensure results are fully generated before reading.
4. **Result Output Phase**: The GPU writes candidate point evaluation results to Unified Memory, which is directly read by the CPU. The Unified Memory space is released after computation.
5. **Error Detection Phase**: The CPU captures exceptions during kernel execution (e.g., out-of-bounds memory access, invalid thread configuration) via CUDA error detection interfaces to ensure robustness of the interaction process.

Kernel Function Encapsulation The core computation logic is encapsulated as a CUDA kernel function, serving as the entry point for GPU parallel computing. The kernel function receives input parameters from the CPU (player states, ball position, core decision-making player identifier, scenario type) and distributes candidate point calculation tasks to threads according to thread/block partitioning rules, ultimately outputting kinematic parameters and validity flags for each candidate point.

3.3 GPU-Side Implementation of Core Algorithms

Atomic computation logic is encapsulated as device-side functions on the GPU to complete end-to-end evaluation of candidate points, ensuring parallel efficiency:

Kinematic Time Prediction for Robots For calculating the time required for a robot to reach a candidate point, a "1D decomposition + 2D merging" approach is adopted:

- **1D Calculation**: Based on robot motion constraints (maximum speed, maximum acceleration/deceleration), the motion time is computed in three phases (acceleration-constant speed-deceleration) to address boundary scenarios such as "initial velocity opposite to target direction" and "insufficient distance to decelerate to stop".
- **2D Merging**: The maximum motion time in the x and y directions is taken as the total arrival time, accounting for the coupling of 2D robot motion and ensuring accurate time prediction.

Translational Trajectory Calculation for the Ball A translational trajectory model for the ball is constructed, considering the segmented effects of rolling friction and sliding friction:

- **Friction Coefficient Adaptation**: Rolling and sliding friction coefficients are adjusted for simulation/real-world scenarios, with the sliding friction coefficient set to a fixed multiple of the rolling friction coefficient to match the ball's motion characteristics in different environments.
- **Displacement and Velocity Calculation**: Based on friction coefficients and initial velocity, the sliding/rolling displacement of the ball at different times is computed to derive the minimum/maximum translational velocity that satisfies the "arrival time-distance" constraint, ensuring the feasibility of passing trajectories.

Safety Verification of Candidate Points Each thread independently completes multi-dimensional legality verification of candidate points, with only valid points marked as such:

1. *Field Boundary Verification*: Check if the candidate point is within the field (with a reasonable boundary margin) to exclude out-of-bounds positions.
2. *Penalty Area Filtering*: Exclude invalid candidate points within the penalty area to comply with competition rules.

3. *Player Validity Filtering*: Exclude candidate points corresponding to the core ball-controlling player or invalid-state players.
4. *Safety Distance Verification*: Define a safety distance threshold based on player size to avoid interception of candidate points by opposing players.

3.4 Scenario Adaptation and Parameter Design

The module achieves simulation/real-world scenario adaptation via a modular parameter management mechanism, with core parameters designed by function category—scenario adaptation is achieved solely through parameter adjustment:

Table 1: Parameter Design for Scenario Adaptation

Parameter Category	Design Principle
Field Parameters	Define field boundaries and penalty area ranges based on standard SSL field dimensions to constrain candidate point space.
Kinematic Constraints	Differentiate maximum speed/acceleration for our/opposing players to ensure accurate kinematic prediction for different roles.
Safety Thresholds	Define safety distance based on player size to balance candidate point usability and anti-interception performance.
Friction Coefficients	Configure rolling/sliding friction coefficients for simulation/real-world scenarios to match actual motion characteristics.

3.5 CPU-GPU Interaction Implementation and Computational Task Allocation

The core design principle of the module is to *"let the GPU do what it excels at, and the CPU do what it is suited for"*. Computational tasks are partitioned based on heterogeneous computing characteristics, with efficient interaction achieved via a standardized workflow:

Computational Task Allocation Logic

– **GPU-Assigned Tasks**: All compute-intensive, parallelizable tasks are handled by the GPU, including:
  1. Generation of coordinates for all candidate points;
  2. Calculation of robot arrival time for each candidate point;
  3. Calculation of ball translational velocity and trajectory for each candidate point;

4. Multi-dimensional safety verification for each candidate point.

  Core Logic: The GPU's many-core architecture is optimized for "large data volume, uniform computation logic, and no dependencies"—maximizing hardware utilization.

- **CPU-Assigned Tasks**: Non-parallel, logic-intensive, or cross-module interaction tasks are handled by the CPU, including:
  1. Data reception and preprocessing from vision/decision modules (e.g., filtering valid player states);
  2. Configuration of GPU computation parameters (e.g., scenario type, core player identifier);
  3. Triggering and synchronization of GPU kernel execution;
  4. Post-processing of GPU results (e.g., scoring and filtering optimal positions, outputting data to the decision layer);
  5. Module error detection and fault tolerance.

  Core Logic: The CPU is optimized for "complex logic, small data volume, and cross-module interaction"—ensuring system flexibility and scalability.

Workflow for CPU-GPU Interaction

1. **Initialization Phase**: The CPU initializes the CUDA runtime environment, allocates Unified Memory space, and defines thread/block dimensions for GPU kernels.
2. **Data Preprocessing and Input**: The CPU retrieves raw player/ball data from the vision module, filters valid data, and writes it to Unified Memory.
3. **Parallel Computation Trigger**: The CPU invokes the kernel function to start GPU parallel computing—meanwhile, the CPU can process other non-dependent tasks (e.g., communication with the decision layer) in parallel.
4. **Computation Synchronization**: The CPU waits for the GPU to complete all candidate point calculations via synchronization interfaces to ensure result integrity.
5. **Result Post-Processing**: The CPU reads GPU-generated candidate point evaluation results from Unified Memory and filters optimal passing/shooting positions using a scoring model.
6. **Resource Release**: The CPU releases Unified Memory space to close the interaction loop for a single computation cycle.

3.6 Key Optimization Highlights

Targeted optimizations were implemented for heterogeneous computing requirements in the RoboCup SSL scenario:

1. **Parallel Granularity Optimization**: Candidate point calculation is delegated to the thread level (each thread processes one candidate point), avoiding resource competition between blocks and maximizing GPU core utilization.

2. **Memory Optimization**: The Unified Memory mechanism reduces CPU-GPU data transmission overhead, with only core computation logic retained on the GPU to minimize GPU memory usage.
3. **Robustness Optimization**: A numerical anomaly filtering logic is designed to avoid module crashes caused by floating-point calculation errors, ensuring stability in extreme scenarios.
4. **Task Partitioning Optimization**: Computational tasks are partitioned based on CPU/GPU hardware characteristics to avoid resource waste (e.g., "GPU executing serial logic" or "CPU executing compute-intensive tasks"), improving overall computing efficiency.

References

1. Huang, Z., et al.: ZJUNlict: RoboCup SSL 2018 Champion Team Paper. In: Robot Soccer World Cup. Springer, Berlin, Heidelberg (2018)
2. Ommer, N., Ryll, A., Geiger, M.: TIGERs Mannheim Extended Team Description for RoboCup 2019. RoboCup Wiki (extended team description), Leipzig, Germany (2019). Accessed 5 April 2019
3. Wu, Y., Zhao, Y., Xiong, R.: ZJUNlict Team Description Paper for RoboCup 2013. Robot Soccer World Cup (2013)
4. Zhao, Y., Xiong, R., Tong, H., Li, C., Fang, L.: ZJUNlict Team Description Paper for RoboCup 2014. Robot Soccer World Cup (2014)
5. Chen, Z., et al.: SRC Extended Team Description Paper for RoboCup 2024 (2024)
6. Wang, Y., et al.: SRC Extended Team Description Paper for RoboCup 2025. Student Innovation Center, Shanghai Jiao Tong University, P.R.China (2025)
7. Wu, Z., et al.: ZJUNlict Extended Team Description Paper. Small Size League of RoboCup 2025. State Key Lab. of Industrial Control Technology, Zhejiang University, Hangzhou (2025)
8. NVIDIA Corporation: CUDA Toolkit Documentation. Version 12.2. NVIDIA Developer Zone (2023). https://docs.nvidia.com/cuda/
9. RoboCup Federation: Small Size League (SSL) Rulebook 2025. RoboCup Official Website (2025). https://www.robocup.org/