RoboCup 2026 Extended Team Description Paper Ri-one

Yuzuru Naito, Shutaro Otake, Bunichi Katsumi,
Yuito Enomoto, Aoi Oka,
Yuto Yoshida, Takumi Banba, Mitsuyoshi Sugaya,
Haruki Sugiyama, Yuki Nakahigashi

College of Information Science and Engineering,
Ritsumeikan University, Iwakura-cho, Ibaraki-shi, Osaka-fu, Japan
Website:https://www.rione.org
{rione.robocup}@gmail.com

Abstract. This paper is about our team's bid to qualify for the RoboCup -2026 soccer small size league tournament. In addition, our team's robot and system are redesigned according to the result of RoboCup2024. Our organization conducts research and analysis in the areas of hardware, circuits, and software.

Keywords: RoboCup · soccer small size league · autonomous robot · engineering education.

1 Introduction

We, Ri-one, are a student organization aiming to be the best RoboCup team in the world, recognized as a project team by the College of Information Science and Technology, Ritsumeikan University. Our team was formed 5 years ago and participated in the world competition for the first time at RoboCup 2022 Bangkok, where we placed 5th, in Bordeaux three years ago, where we placed 3rd, and in Eindhoven the year before last, where we won the world championship. We are currently proceeding with development in preparation for the challenge of Division A. This paper is a continuation of the Extend Team Description Paper for RoboCup 2026, and introduces improvements and new developments compared to the previous version.

2 Hardware

2.1 Dribbler

Unlike last year's model, which absorbed impact via a rotational mechanism pivoting at the drum's base, the new dribbler employs a slot-shaped damper. This design allows for impact absorption through deformation that is not limited to specific translational or rotational axes. We compared three damper materials: silicone tubing (Shore A60), urethane tubing (Shore A70), and 3D-printed TPU. Qualitative observations of ball impact behavior and vibration damping indicated that TPU offered the superior shock absorption. This is attributed to TPU's higher internal loss (hysteresis) compared to rubber-based materials, which effectively dissipates impact energy as heat. Furthermore, 3D printing allows for the optimization of shape, wall thickness, and infill density, providing significant freedom to enhance damper performance.

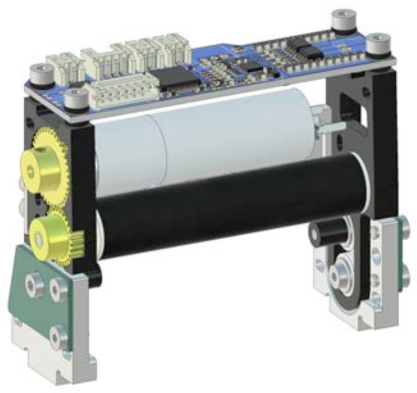

Fig. 1. The new dribbler

2.2 Generative Design

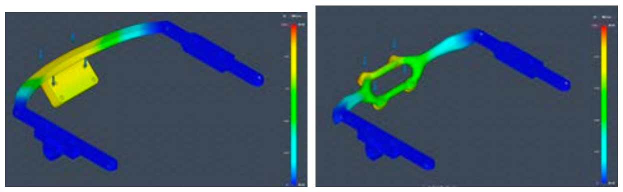

Fig. 2. Camera mount(Left:human, Right:GD)

We utilized Autodesk Fusion's Generative Design (GD) to create components for securing the camera and chassis. GD automatically generates geometries that minimize mass while satisfying input constraints such as load paths, safety factors, and material properties. Analysis of the GD-generated camera mount revealed a maximum displacement of only 0.7 mm, a significant improvement in rigidity compared to the 2.0 mm displacement of the conventional design. Notably, while maintaining the same mass, the GD shape reduced displacement to approximately one-third by optimizing material distribution. This increased stiffness suppresses vibration transmission to the camera, thereby stabilizing image processing. While we explored applying GD to other components, we found that for parts without clear optimization goals (like minimizing displacement), the process was inefficient due to long computation times and complex parameter tuning. Therefore, we limited GD adoption to components where its structural benefits were most pronounced.

2.3 Kicker

Structural Rationale The wheel unit surrounds the kicker, severely restricting available space. To maximize the solenoid's electromagnetic force within this constraint, we adopted a rectangular cross-section coil instead of a conventional cylindrical one. This shape conforms better to the housing, reducing gaps and allowing for approximately 1.4 times the coil volume of a circular design. This increased volume accommodates more windings, theoretically enhancing force generation. Additionally, since the kinetic energy transferred to the plunger depends on the duration of force application, we designed the mechanism to allow for the maximum structurally possible stroke of 46 mm. This extended stroke maximizes acceleration time, ensuring greater energy transfer to the ball.

Prototyping and Evaluation To optimize ball initial velocity, we prototyped seven solenoid variations with different wire diameters, turn counts, and resistance values (see Table 1-1). The goal was to find the optimal balance between maximum allowable circuit current and the time constant.

Table 1. Parameters of Prototyped Solenoids

No.	**Wire Diameter** ϕ (mm)	**Turns** (N) (layers)	**Resistance** R (Ω)
1	0.5	1.5	2.2
2	0.5	2.0	2.6
3	0.5	2.25	2.9
4	0.5	2.5	3.2
5	0.4	2.5	6.1
6	0.4	3.5	8.9
7	0.3	4.5	17.9

We measured plunger speed and ball ejection velocity during discharge from a capacitor at constant voltage. We anticipated that low-resistance designs (Nos. 1–4) would generate high instantaneous current, while high-resistance designs (No. 7) would limit current but secure sufficient ampere-turns (NI).

Table 2. Solenoid Specifications and Experimental Results

No.	**Wire** ϕ (mm)	**Turns** N (layers)	**Resistance** R (Ω)	**Avg. Velocity** (m/s)
1	0.5	1.5	2.2	5.95
2	0.5	2.0	2.6	6.03
3	0.5	2.25	2.9	6.18
4	0.5	2.5	3.2	6.12
5	0.4	2.5	6.1	4.88
6	0.4	3.5	8.9	4.92
7	0.3	4.5	17.9	3.81

Experimental Results and Discussion Experimental results indicated that Design No. 3 (Wire: ϕ0.5 mm, Turns: 2.25 layers) achieved the maximum speed of 6.18 m/s and was selected for the final robot. We analyzed the results as follows:

1. **Linearity vs. Maximum Current:** Thinner wires (ϕ0.4 mm, ϕ0.3 mm; Nos. 5–7) resulted in resistance exceeding 6 Ω. According to Ohm's law ($I = V/R$), this severely limited current. Although force F is proportional to $(NI)^2$, the drop in current I outweighed the benefits of increased turns N, resulting in insufficient speed.
2. **Magnetic Force vs. Time Constant:** Among Nos. 1–4, speed initially increased with the number of turns (from No. 1 to No. 3) due to stronger magnetic flux. However, No. 4 showed a speed decrease. This is because inductance L increases with N^2, raising the time constant $\tau = L/R$. A large τ slows current rise; for a kicker requiring rapid actuation, the current likely failed to peak before the plunger completed its stroke, reducing power.
3. **Conclusion:** Design No. 3 offers the optimal balance for our specifications, achieving a 1.7× speed increase compared to previous years' models.

Mass Production Improvement Solenoid output is highly sensitive to winding alignment and fill factor. To eliminate performance variance between robots, we developed a custom coil winding machine. This device ensures consistent tension and precise pitch control, minimizing resistance errors and ensuring that mass-produced units match the experimental performance of the prototype.

2.4 Chip Bar

Structural Rationale We adopted a 3D-printed chip bar to prioritize weight reduction. To compensate for the lower material strength compared to aluminum, we utilized Structural Analysis in Fusion to identify stress concentration points. We reinforced these areas with ribs and optimized the cross-section. Generative design was also used to verify theoretical optimal shapes, balancing strength and weight.

Optimization Experiment We conducted experiments to optimize two key interactions: the energy transfer from the solenoid plunger to the chip bar, and the contact between the bar tip and the ball. We compared four conditions based on two variables (Table 2-1):

Back Surface Treatment: Smoothing the plunger contact surface via filing and resin coating to reduce friction loss.
Tip Sharpening: Grinding the leading edge to a sharp angle to improve ball penetration.

Table 3. Experimental Conditions for Chip Bars

	Plunger contact surface	**Ball contact surface**
Chip bar 1	Untreated	Untreated
Chip bar 2	Treated	Untreated
Chip bar 3	Untreated	Treated
Chip bar 4	Treated	Treated

Table 4. Experimental Results: Average Flight Distance and Launch Angle

	Average Distance (cm)	**Average Angle** (degree)
Chip bar 1	198.03	38.89
Chip bar 2	206.26	44.44
Chip bar 3	192.26	46.08
Chip bar 4	222.40	47.02

Results and Discussion

- **Tip Sharpening:** Sharpening the tip increased the ejection angle by approx. 8°. The sharp tip penetrates deeper under the ball, shifting the force vector vertically. However, without back surface treatment, this increased angle reduced horizontal flight distance due to energy dispersion.
- **Back Surface Treatment:** Smoothing the contact surface increased flight distance. Reducing microscopic surface irregularities minimized friction loss during impact, improving kinetic energy transfer efficiency.
- **Synergistic Effect:** The combination of tip sharpening and back surface treatment yielded the best performance in both angle and distance. The efficiency gains from the back surface treatment compensated for the horizontal force loss caused by the higher launch angle.

2.5 Wheel Unit

The wheel unit underwent a complete redesign focusing on three improvements: transitioning to direct drive, thinning the motor mount, and enhancing omni-wheel grip.

1. **Direct Drive Mechanism:** Replacing the gear reduction system with a direct drive setup eliminates backlash and simplifies the structure. This is expected to increase top speed and reduce weight by minimizing part count.
2. **Thinned Motor Mount:** To accommodate the larger kicker coil, minimizing unit thickness was critical. We achieved this by embedding a portion of the motor into the mount structure, maintaining rigidity while reducing the profile. Thickness was selectively increased in non-critical areas to further ensure structural integrity.
3. **Omni-wheel Improvements:** We increased the diameters of both the main wheel and the sub-wheels (rollers) and added more sub-wheels to improve grip. Larger sub-wheels also reduce friction in the transverse direction. Structurally, we offset the sub-wheel axes to avoid interference while maintaining bearing strength.

These changes resulted in a slimmer unit with higher torque transfer efficiency, higher speed, and zero backlash.

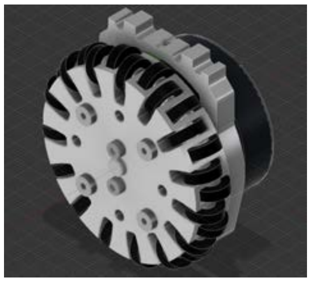

Fig. 3. New Wheel Unit

2.6 Motor Driver

To accommodate higher battery voltage (50 V) and increased power, we consolidated the previous multi-board driver system into a single board featuring four STSPIN32G4 microcontrollers[1]. This centralization minimizes high-voltage wiring, significantly reducing short-circuit risks. We upgraded MOSFETs and optimized the thermal design to prevent runaway. Communication was switched from CAN to UART to isolate the driver from the main bus, preventing overload and simplifying wiring. The single-board design also allows for simultaneous reprogramming of all MCUs, improving maintainability.

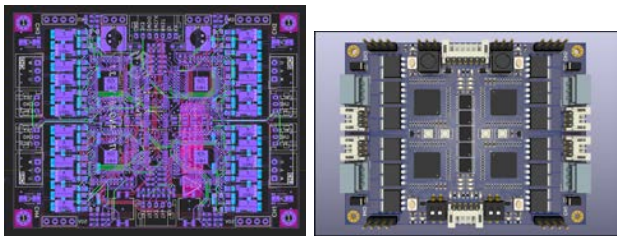

Fig. 4. Motor Driver

2.7 Voltage Booster

To drive 100W motors efficiently, we implemented a boost circuit raising the 25 V battery voltage to 50 V. Built with an Analog Devices MAX15158 and STM32H523, this 2-channel system delivers up to 1 kW (50 V/20 A) with active phase balancing and overcurrent protection. This board functions as a central power hub, supplying 50 V, 25 V, 10 V, 6.5 V, and 3.3 V to the entire robot. It employs forced air cooling (40 mm fans) and uses board-to-board terminals to connect to the motor driver, eliminating cable-related safety risks.

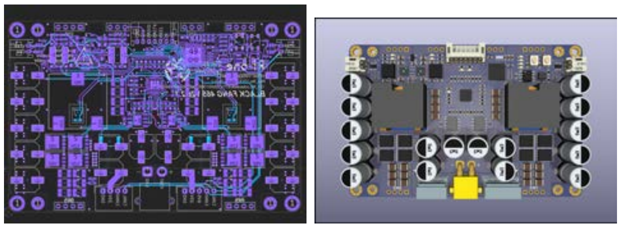

Fig. 5. Voltage Booster

2.8 IMU and Velocity Estimation

The robot is equipped with a BMI270 IMU for self-velocity estimation. The BMI270 was selected for its superior precision (±0.4% error), offering less than 40% of the error margin of the previously used BNO055. To process the accelerometer data, we first remove the startup offset (average of 1,000 samples) and apply a digital low-pass filter. Experimental analysis of forward motion data (Fig. 6) revealed that a 3.5 Hz filter caused excessive phase delay. Therefore, a 10 Hz cutoff was selected to balance noise reduction with response speed.

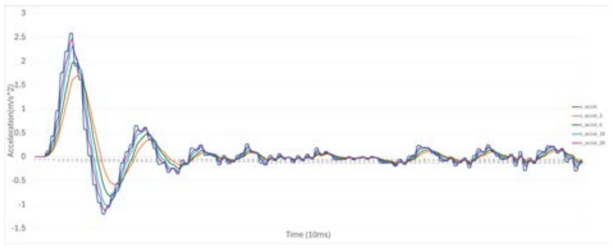

Fig. 6. Aircraft x-axis acceleration and filtered acceleration

Subsequently, sensor fusion was performed using an Extended Kalman Filter (EKF) to combine the filtered acceleration data with angular velocity data from the motor encoders. Figure 7 compares the ground truth velocity, single-sensor estimates, and the fused result. Figure 5 shows the average error.

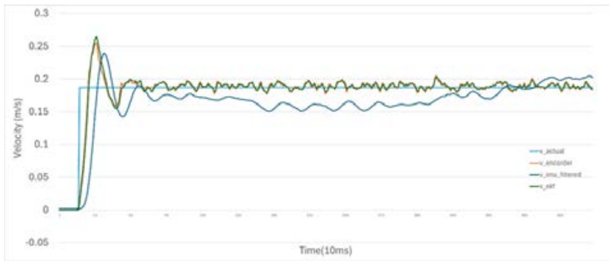

Fig. 7. Comparison of actual speed and speed estimates from each method

Table 5. Comparison of Average Velocity and Absolute Error

	v_{actual}	v_{encoder}	$v_{\text{imu\_filtered}}$	v_{ekf}
Average (m/s)	0.1870	0.1880	0.1688	0.1881
Absolute Error (m/s)	0.0000	0.0010	0.0182	0.0011

From Figs. 7 and 5, three key points are observed:

1. **Acceleration Integration Error ($v_{\text{imu\_filtered}}$):** Velocity derived from integrating acceleration exhibits significant phase delay, signal attenuation, and steady-state error compared to the ground truth (v_{actual}). This is attributed to the delay from the 10 Hz filter and integration drift caused by sensor bias.
2. **EKF and Encoder Dominance:** The EKF estimate (v_{ekf}) closely aligns with the encoder value (v_{encoder}). This results from tuning the observation

noise covariance matrix R to trust the encoder more than the accelerometer. The filter correctly prioritizes the stable encoder data over the noisy accelerometer signal.

3. Evaluation Conditions: The experiment was conducted on a high-grip carpet where tire slip was negligible. Under these conditions, the encoder alone was highly accurate, masking the benefits of sensor fusion. Increasing the accelerometer's weight (R) in this specific scenario would likely degrade accuracy due to the sensor's divergence.

Future work will involve fine-tuning covariance weights, testing at competition speeds, and verifying performance on various surface textures.

3 Software

3.1 RAVEN: Integrated Strategy and Control System

Previously, our software architecture comprised three independent components: RACOON-AI for strategic decision-making, RACOON-MW as open-source middleware, and RACOON-controller for robot control. While this modular design facilitated data logging and independent operation, it introduced challenges in maintainability and inter-process communication compatibility.

This year, we have developed **RAVEN**, a unified strategy and control system. By consolidating all modules into a single Java-based application, we have minimized inter-process communication latency and significantly improved processing speed. This monolithic architecture reduces the overhead associated with network-based message passing while maintaining modularity through well-defined internal interfaces.

3.2 Automatic Robot Discovery and IP Management

In SSL competitions, managing IP addresses for multiple robots presents operational challenges. Conventional approaches require fixed IP addresses (e.g., within the 192.168.0.0/24 subnet), necessitating manual configuration on the host PC and unique SD card images for each robot when using embedded systems such as Raspberry Pi with Wi-Fi communication. This approach is labor-intensive and error-prone during robot replacement or maintenance.

We propose an automatic robot discovery and identification system that dynamically resolves IP addresses and associates them with robot IDs, eliminating the need for manual configuration.

IP Address Management Table Both RAVEN and the latest version of RACOON-MW implement a discovery protocol based on multicast UDP packets. Each robot broadcasts a *Discovery packet* containing its robot ID upon startup and at regular intervals. The host system maintains an IP address management table that maps robot IDs to their dynamically assigned IP addresses.

This design enables robots to obtain IP addresses via DHCP, allowing the use of identical SD card images across all robots. The benefits of this approach include:

– **Simplified Deployment:** A single SD card image can be mass-produced and deployed to all robots without modification.
– **Operational Flexibility:** Robot replacement requires no reconfiguration on the host system.
– **Reduced Configuration Errors:** Automatic discovery eliminates manual IP address assignment mistakes.

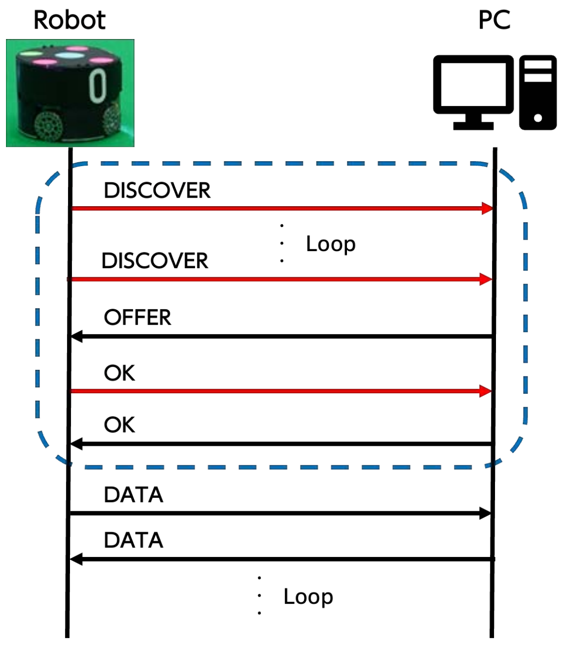

Fig. 8. Robot communication system architecture. Robots broadcast Discovery packets via multicast, and the host system maintains an IP address management table for dynamic address resolution.

Automatic Robot ID Configuration When using identical SD card images, robot IDs must be determined by external hardware. We employ a DIP switch mounted on each robot's PCB, connected to GPIO pins on the embedded controller. This configuration allows setting robot IDs from 0 to 15 (4-bit binary encoding), providing sufficient capacity for SSL team sizes while maintaining hardware simplicity.

The robot ID is read during the boot sequence and used in all subsequent Discovery packet broadcasts. This approach ensures that robot identification is determined by physical hardware configuration rather than software settings, preventing ID conflicts and simplifying field setup procedures.

3.3 Path Planning: Introduction of Informed RRT*

Last year, our team adopted **RRT* (Rapidly-exploring Random Tree Star)** as the core path planning algorithm. RRT* guarantees *asymptotic optimality*, ensuring the path converges to the optimal solution as the number of iterations approaches infinity. However, in the highly dynamic environment of the RoboCup SSL, the convergence speed of standard RRT* proved to be a bottleneck.

Conventional RRT* continues to sample randomly from the entire configuration space even after an initial feasible path is found. This results in wasted computational resources on areas where the cost is clearly higher than the current best solution. To address this inefficiency, we introduced **Informed RRT***[3] this year.

Methodology: Informed RRT* Informed RRT* improves convergence speed by restricting the search space to an "admissible heuristic region" once an initial solution is found.

Mathematical Formulation Let $x_{start} \in X$ be the start position, $x_{goal} \in X$ be the goal position, and c_{best} be the cost of the current best path (Euclidean distance). The theoretical minimum cost c_{min} is defined as:

$$c_{min} = ||x_{start} - x_{goal}||_2 \tag{1}$$

The subset of the search space X containing points x capable of improving the current solution is defined as $X_{\hat{f}}$:

$$X_{\hat{f}} = \{x \in X \mid ||x_{start} - x||_2 + ||x - x_{goal}||_2 \leq c_{best}\} \tag{2}$$

This inequality describes a **prolate hyperspheroid** (an ellipse in 2D space). Informed RRT* generates new samples x_{new} only from within this ellipsoidal region.

Ellipsoidal Sampling To sample uniformly within the ellipse, we transform samples from a unit circle. A point x_{ball} in the unit circle is mapped to x_{inf} in the ellipse via:

$$x_{inf} = C \cdot L \cdot x_{ball} + x_{center} \tag{3}$$

Where L is the scaling matrix (determining major/minor axes), C is the rotation matrix (aligning with the start-goal axis), and x_{center} is the ellipse center.

Implementation Details

Algorithm Flow The path planning process is implemented in the `InformedRRTstar` class. The flow is presented in Algorithm 1. The algorithm dynamically switches its sampling strategy:

1. **Global Sampling:** When $c_{best} = \infty$ (initial phase), it explores the entire field.
2. **Ellipsoid Sampling:** Once a path is found ($c_{best} < \infty$), it restricts search to the ellipsoidal region.

Specifics and Smoothing Our Java implementation includes specific optimizations:

– **Hybrid Sampling (`sampleRandomPoint`):** This method branches based on solution existence. The ellipsoidal sampling uses `Math.atan2` for rotation and scaling based on $a = c_{best}/2$ and $b = \sqrt{a^2 - (c_{min}/2)^2}$.
– **Rewiring:** The neighbor search radius (`rewireRadius`) is fixed at 2.0× the step size to balance load and optimality.
– **Path Smoothing (`smoothPath`):** A post-processing step using a greedy approach removes redundant intermediate nodes, ensuring smooth trajectories suitable for robot motion control.

3.4 Trajectory Optimization: Timed Elastic Band

Following the path generation by Informed RRT*, we employ the **Timed Elastic Band (TEB)** algorithm to generate a time-optimal velocity profile. This algorithm ensures adherence to kinodynamic constraints—such as velocity and acceleration limits—while maintaining obstacle avoidance.

Algorithm Overview The TEB algorithm iteratively optimizes the trajectory through two main processes: *Auto-Resizing*, which adjusts the number of discrete nodes, and *Numerical Optimization* using the Levenberg-Marquardt method.

State Representation

Hypergraph Structure In our implementation, the trajectory is modeled as a hypergraph where spatial poses and temporal intervals alternate in a chain-like structure:

$$\mathbf{P}_0 \xrightarrow{\Delta T_0} \mathbf{P}_1 \xrightarrow{\Delta T_1} \mathbf{P}_2 \cdots \xrightarrow{\Delta T_{n-1}} \mathbf{P}_n \tag{4}$$

Here, the nodes and edges are defined as follows:

– **Nodes (Spatial State):** $\mathbf{x}_i = (x_i, y_i, \theta_i)$. These represent the waypoints the robot must traverse.
– **Edges (Temporal State):** ΔT_i. This represents the time duration required to move from node $\mathbf{P}_i$ to $\mathbf{P}_{i+1}$.

Algorithm 1 Informed RRT* Path Planning Algorithm

```
1: **Input:** $x_{start}$, $x_{goal}$, $X_{free}$ (obstacle-free space), $maxIter$
2: **Output:** Optimized path from $x_{start}$ to $x_{goal}$
3:
4: Initialize tree $T$ with root node $x_{start}$
5: $c_{best} \leftarrow \infty$    ▷ Cost of current best path
6: $c_{min} \leftarrow ||x_{start} - x_{goal}||_2$    ▷ Theoretical minimum cost
7:
8: for $i = 1$ to $maxIter$ do
9:     if $c_{best} = \infty$ then
10:        $x_{rand} \leftarrow$ SampleGlobal($X_{free}$)    ▷ Global sampling
11:    else
12:        $x_{rand} \leftarrow$ SampleEllipsoid($c_{best}$, $c_{min}$, $x_{start}$, $x_{goal}$)    ▷ Ellipsoidal sampling
13:    end if
14:
15:    $x_{nearest} \leftarrow$ Nearest($T$, $x_{rand}$)
16:    $x_{new} \leftarrow$ Steer($x_{nearest}$, $x_{rand}$, $stepSize$)
17:
18:    if CollisionFree($x_{nearest}$, $x_{new}$) then
19:        $X_{near} \leftarrow$ Near($T$, $x_{new}$, $rewireRadius$)
20:        AddNode($T$, $x_{new}$)
21:
22:        // Choose best parent
23:        $x_{min} \leftarrow x_{nearest}$
24:        $c_{min} \leftarrow$ Cost($x_{nearest}$) $+||x_{nearest} - x_{new}||_2$
25:        for $x_{near} \in X_{near}$ do
26:            $c \leftarrow$ Cost($x_{near}$) $+||x_{near} - x_{new}||_2$
27:            if $c < c_{min}$ and CollisionFree($x_{near}$, $x_{new}$) then
28:                $x_{min} \leftarrow x_{near}$
29:                $c_{min} \leftarrow c$
30:            end if
31:        end for
32:        AddEdge($T$, $x_{min}$, $x_{new}$, $c_{min}$)
33:
34:        // Rewire tree
35:        for $x_{near} \in X_{near}$ do
36:            $c \leftarrow$ Cost($x_{new}$) $+||x_{new} - x_{near}||_2$
37:            if $c <$ Cost($x_{near}$) and CollisionFree($x_{new}$, $x_{near}$) then
38:                RemoveEdge($T$, Parent($x_{near}$), $x_{near}$)
39:                AddEdge($T$, $x_{new}$, $x_{near}$, $c$)
40:            end if
41:        end for
42:
43:        // Update best cost if goal is reached
44:        if $||x_{new} - x_{goal}||_2 <$ threshold then
45:            $c_{best} \leftarrow \min(c_{best}$, Cost($x_{new}$) $+||x_{new} - x_{goal}||_2)$
46:        end if
47:    end if
48: end for
49:
50: return SmoothPath(ExtractPath($T$, $x_{start}$, $x_{goal}$))
```

Parameter Vector To perform mathematical optimization, we construct a single parameter vector $\mathbf{\Theta}$ by concatenating the pose and time variables. In the `extractParameters` method of our optimizer, $\mathbf{\Theta}$ is defined as:

$$\mathbf{\Theta} = [\underbrace{x_1, y_1, \theta_1}_{\mathbf{P}_1}, \underbrace{\Delta T_0}_{\text{time}}, \underbrace{x_2, y_2, \theta_2}_{\mathbf{P}_2}, \underbrace{\Delta T_1}_{\text{time}}, \cdots]^T \tag{5}$$

Note that $\mathbf{P}_0$ is typically fixed as the current robot state.

Auto-Resizing Since the initial node distribution depends on the RRT* output, we dynamically adjust the trajectory resolution based on physical constraints via the `autoResize()` method.

- **Refining:** If a time interval ΔT_i exceeds a specific threshold, intermediate poses are inserted to subdivide ΔT_i. This increases resolution for finer control.
- **Coarsening:** If ΔT_i is below a lower threshold, the node is removed, and the adjacent intervals are merged. This reduces computational load for faster processing.

Levenberg-Marquardt Optimization The core optimization is performed in the `optimize()` method using the Levenberg-Marquardt (LM) algorithm, a standard technique for non-linear least squares problems.

Objective Function We define the objective function $F(\mathbf{\Theta})$ as the sum of squared residuals corresponding to various constraints:

$$F(\mathbf{\Theta}) = \sum_k r_k(\mathbf{\Theta})^2 = \|\mathbf{r}(\mathbf{\Theta})\|^2 \tag{6}$$

where $\mathbf{r}$ is the residual vector. The residuals r_k, calculated in `calculateResiduals()`, represent the violation magnitude of each constraint weighted by w:

- **Obstacle Constraint:** $r_{obs} = \max(0, \text{minDist} - \text{actualDist}) \times w_{obs}$. Penalizes intrusion into the safety margin.
- **Velocity Constraint:** $r_{vel} = \max(0, v_{curr} - v_{max}) \times w_{vel}$.
- **Acceleration Constraint:** $r_{acc} = \max(0, a_{curr} - a_{max}) \times w_{acc}$.
- **Time Optimality:** $r_{time} = \Delta T_i \times w_{time}$. Minimizes total travel time.
- **Distance Constraint:** $r_{dist} = |\mathbf{P}_i - \mathbf{P}_{i-1}| \times w_{dist}$. Maintains spatial consistency between nodes.

Update Rule To find the optimal $\mathbf{\Theta}$, the LM algorithm calculates the update step $\boldsymbol{\delta}$ by solving:

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\boldsymbol{\delta} = -\mathbf{J}^T\mathbf{r} \tag{7}$$

where $\mathbf{J}$ is the Jacobian matrix of $\mathbf{r}$, and λ is the damping factor.

- **Large λ (Gradient Descent):** The equation approximates $\lambda\mathbf{I}\boldsymbol{\delta} \approx -\mathbf{J}^T\mathbf{r}$, ensuring stable convergence in the direction of the gradient.
- **Small λ (Gauss-Newton):** The equation approximates $(\mathbf{J}^T\mathbf{J})\boldsymbol{\delta} = -\mathbf{J}^T\mathbf{r}$, providing rapid convergence by considering curvature.

Gain Ratio and Step Acceptance We evaluate the quality of the update candidate $\boldsymbol{\delta}$ using the gain ratio ρ:

$$\rho = \frac{\text{Actual Reduction}}{\text{Predicted Reduction}} = \frac{F(\boldsymbol{\Theta}) - F(\boldsymbol{\Theta} + \boldsymbol{\delta})}{\text{Predicted Reduction}} \tag{8}$$

To avoid computationally expensive matrix multiplications ($\mathbf{J}^T\mathbf{J}$) for the predicted reduction, we derive an approximation by substituting the update equation:

$$\begin{aligned} \text{Pred. Red.} &= \boldsymbol{\delta}^T(\lambda\boldsymbol{\delta} \underbrace{-\mathbf{J}^T\mathbf{r}}_{\text{Substituted}}) \\ &= \boldsymbol{\delta}^T(\lambda\boldsymbol{\delta} + \mathbf{J}^T\mathbf{J}\boldsymbol{\delta} + \lambda\boldsymbol{\delta}) \\ &= 2\lambda\|\boldsymbol{\delta}\|^2 + \|\mathbf{J}\boldsymbol{\delta}\|^2 \end{aligned} \tag{9}$$

If ρ exceeds a threshold, the update is accepted ($\boldsymbol{\Theta}_{new} \leftarrow \boldsymbol{\Theta} + \boldsymbol{\delta}$), and λ is decreased (trust region expanded). Otherwise, the update is discarded, and λ is increased to enforce a more robust step.

3.5 Ball Detection

In recent years, object detection technologies using image processing and machine learning have achieved remarkable development. In SSL, an increasing number of teams have been equipping their robots with cameras for ball detection. The motivation behind this is to prepare for situations where the ball enters the blind spot of SSL Vision or when Vision-related problems occur. While deep learning models for depth estimation have been proposed as a technique to estimate the distance to objects, their implementation on SSL robots is likely to be hindered by computational load and the integration of sensors.

Therefore, we implemented a simple algorithm to estimate the distance to a ball of known size using a monocular camera. This method is designed as a backup to be used when Vision encounters problems. If the focal length and object size are known, the distance can be calculated with millimeter precision from a single image. This method requires camera calibration and undistortion. We obtained parameters using OpenCV functions[4,5].

We approximate the camera as an ideal pinhole camera model. Let us denote a point in the world coordinate system extended to the camera coordinate system as $\mathbf{P}^h = (X, Y, Z, 1)^T$ and the corresponding point on the image as $\mathbf{p}^h = (x, y, 1)^T$. The projection relationship from $\mathbf{P}^h$ to $\mathbf{p}^h$ is defined as equation (10) and (11).

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{10}$$

$$s\mathbf{p}^h = \mathbf{K}(\mathbf{R}|\mathbf{t})\mathbf{P}^h \tag{11}$$

Here, $\mathbf{K}$ is the camera intrinsic parameter matrix, which includes the focal lengths f_x, f_y and the principal point coordinates (c_x, c_y) of the image coordinate system. $\mathbf{R}$ is the rotation matrix. $\mathbf{t}$ is the translation vector from the

world coordinate system to the camera coordinate system. s is the scale factor of the image coordinate system.
The focal length is approximated by Equation (12) for cameras with nearly square pixels. The distance to the object D[mm] is expressed as Equation (13) using the diameter d[px] measured on the image.

$$f_{px} \approx f_x = f_y \tag{12}$$

$$D = \frac{2f_{px} \cdot r}{d} \tag{13}$$

Here, r[mm] represents the physical radius of the ball.

4 Conclusion

In this paper, we have discussed and introduced our new improvements. As mentioned, our team has been focusing on developing both our software system and the hardware of our machines, and these improvements are expected to contribute to a strong performance in Division A. Although some of them are still under development, we will continue our efforts to achieve success in the competition.

References

1. STMicroelectronic, " STSPIN32G4 datasheet", STMicroelectronic Documentation, `https://www.st.com/resource/en/datasheet/stspin32g4.pdf`, Accessed Jan. 27, 2026
2. OpenCV, "Basic Operations on Images", OpenCV Documentation, Jan.28, 2025, `https://docs.opencv.org/4.x/d3/df2/tutorial_py_basic_ops.html`, Accessed Jan. 27, 2026
3. J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic", 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2997-3004, `https://ieeexplore.ieee.org/document/6942976`, Accessed Jan. 27, 2026
4. OpenCV, "Camera Calibration", OpenCV Documentation, Jan.28, 2025, `https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html`, Accessed Jan. 27, 2026
5. OpenCV, "calib3d — Camera Calibration and 3D Reconstruction", OpenCV Documentation, Jan.28, 2025, `https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html`, Accessed Jan. 27, 2026