NAMeC - Team Description Paper Small Size League RoboCup 2026 Application of Qualification in Division B

T.W. Menier, A. Calugi, P. Félix, C. Godinat, C. Labbé, J. Lindois, C.A. Vlamynck, M. Vulliez, T. Templier–Bourda

IUT - Université de Bordeaux, Gradignan, France
ENSEIRB-MATMECA, Bordeaux INP, Talence, France
namec.team@gmail.com (corresponding author)

Abstract. This paper outlines the recent advancements and technical refinements of NAMeC, a French robotics team competing in the RoboCup Small Size League. Building upon our transition to a student-led organization, this year's developments focus on increasing system robustness and coordination complexity. On the hardware side, we present a redesigned two-piece protective robot shell optimized for durability and ease of maintenance. In electronics, we detail the transition to a CAN bus architecture to improve internal communication reliability. Finally, we introduce new control techniques to correctly follow the rules during a match, and to perform formations.

Keywords: RoboCup, Small Size League, CAN bus, Modular Design, Quadratic Programming, Formation Control, Discrete-time consensus, Autonomous Robots

1 Introduction

Since our debut in the RoboCup Small Size League in 2018, NAMeC has participated a total of five times at the international RoboCup competitions. A pivotal moment in our history occurred in 2024, when the team officially transitioned to a fully student-led organization. While the 2025 season was a period of adaptation to this new structure, the current cycle represents a milestone of consolidation. This year's developments are driven by the need for increased reliability and sophisticated team play. In the mechanical domain, we have replaced our previous shell with a two-part modular shell. This new design features a fixed lower section and an independent upper assembly secured by fastening notches, whose impact resistance has been improved and easily prototyped via 3D printing using PLA. A new CAN bus system is being tested to enhance data integrity when transmitting command packets to the motor cards. Finally, we present progress in our decision software, introducing quadratic program based rule-following and relative robot formations. These improvements allow the team to maintain complex spatial configurations while strictly adhering to match rules through optimization.

2 Redesign of the robot's shell

During RoboCup 2024 at Eindhoven, we have localized different issues in the mechanical conception of the robot's shell. The two main issues are related to the fixation of the hood and its mechanical integrity. This fixation problem gives the shell some liberty to rotate around the Z axis, and thus the angle detected by the vision system is different from the actual robot's angle. This error makes interception of a moving ball difficult. Moreover, this misalignment disrupts passes and shooting of our robots. The second problem was the deterioration of the robot shells, they were not strong enough to withstand the repeated impacts during transport and matches, which would even leave bits of the shell on the field.

2.1 Alignment of the shell

The alignment problem comes directly from the assembly of the shell, which is the reason it creates a degree of liberty on the Z axis. This allows the shell to rotate because of vibrations or impacts when the robot is moving, until attaining a mechanical stop, which is showcased in figure 1.

Fig. 1: The two mechanical stops of the robot's shell, without having moved the robot from its location.

After multiple 3D-printed iterations to resolve this problem, a new version has been engineered following a two-piece architecture : a lower part aligned with the robot using cavities (showcased in red in figure 2), and an upper part, easily removable to facilitate maintenance, that has the corresponding notches (see figure 3).

Both parts are easily assembled together and interlocking is ensured by friction. To avoid the alignment problems cited earlier, the blue shapes shown on figure 2 were designed to match the shape of the chassis' aluminum plate of our robot, ensuring a reproducible positioning and removing any parasite degree of

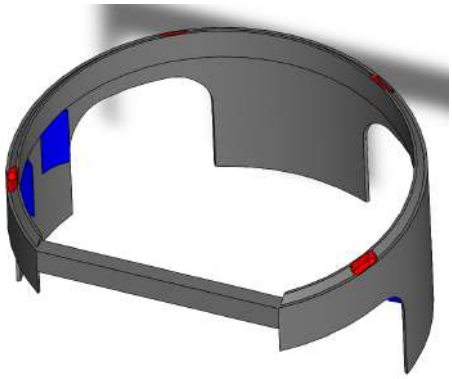

Fig. 2: Lower part of the shell, with cavities in red, and blue shapes representing stop notches that match the intermediary aluminum plate.

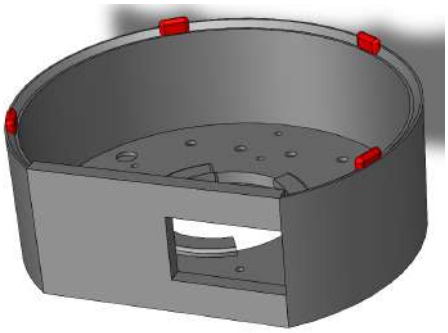

Fig. 3: Upper part of the shell, upside down, with the notches shown in red.

freedom. This solution allows us to make the lower part perfectly supportive from the rest of the robot. With the fixation eliminating any possible misalignment of the shell, ensuring correct orientations of the robot during a match.

2.2 Integrity of the shell

The principal cause of the weakness of the 3D-printed shell was the material used for printing and the printing orientation. We used PLA, which is not very resistant to impacts, as mentioned by the database in [4]. Deterioration was mainly caused by the printing orientation. Because the material was printed vertically, it created the shell into multiple horizontal layers that are stacked together. In the future, we'd like to try to change the printing orientation to improve resistance, as well as using a different printing material, such as ABS, that has better results to the Izod impact strength [13].

Separating the robot's cover in two parts makes it possible to not have to replace the totality of the cover in case of damage. Observations on the previous generation of 3D-printed covers has shown that most of the breakages were occurring around the lower part of the (full) cover, because of the thickness of this part. In next versions, we will try increasing its thickness to further improve resistance.

3 Improving internal communication

In the current generation of our robots, electromagnetic interference generated by the motors and the kicker coil are a big concern. These interferences lead to packet corruption on the SPI bus, which are currently mitigated using a Cyclic Redundancy Check (CRC) code inside each packet transmitted to ensure it is valid. Recall that our architecture is comprised of a main control board connected to the 4 motor cards using SPI cables to transmit information. While the CRC check is sufficient to drive robots, there is still a high number of corrupted packets during transmission.

In 2023, we mentioned the use of Trinamic components to replace our motors cards [7]. Past members of the team had already developed a motor card that integrates the TMC4671 chip, and have produced a high quantity of them, but did not finish integrating it in the robot due to electromagnetic noise. The particularity of the motor card created is that it does not contain a CPU that can be reprogrammed, only the Trinamic chip, which is only meant to be configured when it is powered, by sending commands to it. In 2025, a group of computer science engineering students from the ENSEIRB-MATMECA engineering school have worked on writing the code and calibration required to integrate this new motor card into our robot [3]. While the project was a success when the motor cards were outside the robot, electromagnetic noise still made the motor cards practically unusable.

Thus, in preparation for the future integration of Trinamic motor cards, we are currently evaluating an alternative communication architecture. To ensure

correct data transmission to the motor cards, this communication has to be error-free because the Trinamic motor cards directly use the data of the serial communication. Unfortunately, the Trinamic chip integrated is unable to check for errors. The approach chosen to solve this problem consists of introducing an SPI-to-CAN conversion stage on the existing SPI communication line. This proposed architecture enables communication between the mainboard and all motor boards through an SPI–CAN–SPI chain (shown in figure 4), which theoretically improves robustness against electromagnetic disturbances, while remaining compatible with the current hardware. The goal of this experiment is to determine if the usage of the CAN bus is an advantage or a disadvantage looking at the robustness, data transmission speed, and finally number of cables.

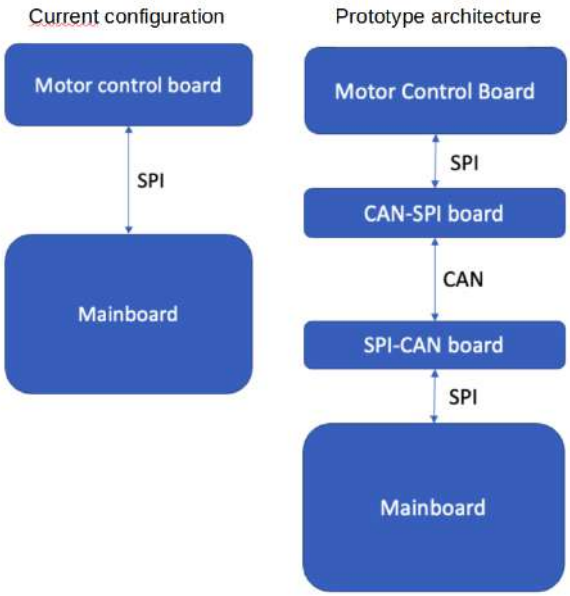

Fig. 4: Communication diagram of the current architecture (left) and the proposed architecture (right) with the SPI-CAN-SPI conversion chain.

4 Relative formations using position consensus

Until now, the game play of our robots consisted of individual behaviors. Last year, we presented a novel strategy manager, named "BigBro" [2] to orchestrate multiple strategies at once. Its usage has proven to be difficult to use for developers and not very efficient as a whole. Instead of coordinated singular behaviors, we propose the use of discrete-time consensus to drive a flock of robots together,

without relying on absolute positioning. An example usage is a triangle formation following the attacker with the ball, so that it may have open possibilities for passes.

The concept of consensus has been extensively studied in [11,10]. Usage of consensus for flocking has been shown for simple systems with a single input and output [9] and MIMO systems [6]. We refer to consensus as a collection of agents influencing each other to converge to a common value. The influence between agents is modeled using the classical definitions of graph theory in (1), (2) and (3).

$$G = (V, E) \tag{1}$$

$$A = [a_{ij}] \tag{2}$$

$$N_i = \{j \in V \mid a_{ij} \neq 0\} \tag{3}$$

Here, G is a directed graph, with the set of vertices $V = \{1, 2, \cdots, n\}$ and edges $E = \{\{x, y\} \mid x, y \in V \ and \ x \neq y\}$, disallowing an agent to be linked to itself. A is the adjacency matrix of the G such that if there is a link from i to j, then $a_{ij} = 1$. Lastly, we need the definition of neighbors of an agent N_i given by (3).

To use a consensus controller for formation control, consider that agents want to converge to a common 2D position, but with a relative offset $d_{ij} \in \mathbb{R}^2$ that agent i should maintain from agent j. The (x, y) position of an agent at discrete time step[1] k is a 2-dimensional vector $x_i[k] \in \mathbb{R}^2$. With this formulation, we modify the discrete-time controller of [11] resulting in equation (4).

$$x_i[k+1] = x_i[k] + \epsilon \sum_{j \in N_i} x_j[k] - x_i[k] - d_{ij} \tag{4}$$

$$s.t \quad 0 < \epsilon < \frac{1}{\Delta} \tag{5}$$

Δ represents the maximum out-degree[2] of the graph. ϵ is a step-size that defines the speed at which consensus should converge. Constraint (5) ensures convergence of the original discrete-time consensus controller. Section 4.4 of [9] gives a informal proof on the rationale for this constraint in discrete-time. Unfortunately, the addition of the offset term d_{ij} means that proof of convergence from [11] does not apply. Thus, consensus might not converge, leading to the formation drifting over time. We try to provide an informal proof of divergence for this specific controller.

In directed graphs, divergence of consensus may occur if two agents i and j form a strongly connected component, meaning $(i, j) \in E$ and $(j, i) \in E$, and that their relative offsets do not sum to 0. Another way to phrase this is to

[1] Other papers use parenthesis with letter k to describe discrete time steps. We choose to use the notation of [10] as the brackets make it clearer that we use discrete time steps, in the opinion of the author.

[2] In-degree can also be used to model the opposite influences.

saythat every pair of agent (i, j) forming a strongly connected component should respect condition (6).

$$d_{ij} - d_{ji} = 0 \tag{6}$$

A real-life example makes the problem apparent : if a person A tries to hug you, but you refuse, you will continuously run away from said person. Person A tries to reduce this offset from you ($d_{A,you} = 0$), while you try to avoid the hug ($d_{you,A} > 0$). Consensus can never be achieved, thus the formation drifts away. A simulation was conducted with condition (6) and has proven to be sufficient, with the results presented in figure 5 for a 2D system.

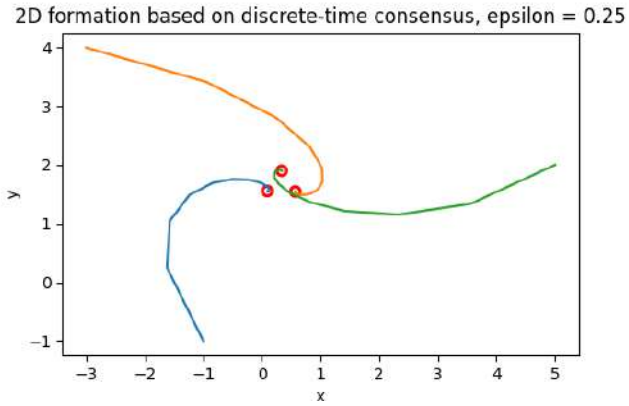

Fig. 5: Simulation of a formation realized using discrete-time consensus, with offsets applied. The influence graph is connected, and each agent has only one neighbor.

This consensus controller enables the modeling of any type of formation, assuming constraints (5) and (6) are respected. The main advantage of this method is that the formation is formed in a relative manner, independently of the current location of each agent. The target usage of this controller is to automatically drive a formation around an autonomous leader. Extracting the second term of the consensus controller in (4), the following speed controller (7) can be formed to achieve the formation [9], simply because the consensus is based on their positions.

$$\dot{x}_i[k] = \epsilon \sum_{j \in N_i} x_j[k] - x_i[k] - d_{ij} \tag{7}$$

Application of this speed controller is showcased in our qualification video [15], and is the one used for simulation results of figure 5. Lastly, one advantage of this controller is that it may be used deported for distributed computation on each robot's processor, which is detailed in [9].

5 Optimization-based rule following

During a SSL match, it is mandatory to follow the rules enforce by the automatic referees, due to the risk of having multiple robots removed from the game, evidently putting us at a disadvantage. Previously, our team incorporated these rules inside the behaviors implemented for our robots, but this method presents a major problem : new behaviors must be thoroughly reviewed to ensure they respect rules, limiting the possibility to create freely without limits. In the field of control theory, control barrier functions (or CBFs) [1] have been used for command control [5] and obstacle avoidance [8]. As we currently control our robots using speed inputs, a speed-based guarding mechanism is introduced, based on these functions and the resolution of a quadratic program (or QP) to change the speed commands [8,9]. Ultimately, the goal is to lift any restrictions on the design of strategies, delegating rule-enforcement to the guarding mechanism. From what we've searched, no team have proposed the usage of CBFs in the league. We also used the TDP search tool [14] across all leagues of the RoboCup, and could not find any either. While the principles that will be introduced for our use-case are well detailed in [8], we'd like to present its results without the generalization of Lie algebra. This is also necessary to easily integrate some results of [9].

5.1 Control barrier functions as constraints

Let us define what is a CBF, simplified to the case of 2-dimensional speed control[3]. Consider the system $\dot{x} = f(x)+g(x)u\ ,\ \ x \in \mathbb{R}^n, u \in \mathbb{R}^m$, i.e. a speed-driven robot. In this system, the set C will represent the set of positions that are safe for it to attain. Forbidden (or unsafe) locations can be considered as virtual obstacles that the robot should not collide with. If we can define a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ such that, if $h(x) < 0$, a collision occurs, then the safe set of positions can be modeled by (8).

$$C := \{x \in \mathbb{R}^n \mid h(x) \geq 0\} \tag{8}$$

The main takeaway of [1] is to use the forward invariance of a set. This property ensures that, if a value $x = x_0$ starts inside the safe set C, then x will never leave the set C. A graphical representation is given in figure 6. If the controller respects the condition cited in (9), then $h(x)$ is called a *Control Barrier Function* for the set C, and thus ensures its forward invariance property.

$$\dot{h}(x) + \alpha(h(x)) \geq 0 \tag{9}$$

α is referred to as a K-class function [16], meaning it is strictly increasing and $\alpha(0) = 0$. This condition may be reduced to a coefficient [9], i.e. $\alpha(x) = \alpha x$, if and only if $\alpha > 0$. Theoretically, this definition breaks the second condition

[3] [1] presents a generalized form of control barrier functions using Lie algebra. These results have been simplified to our case for ease of understanding.

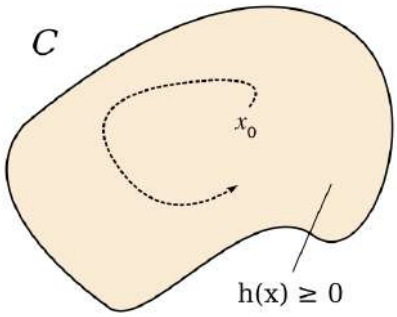

Fig. 6: Representation of a safe set C defined with the control barrier function $h(x)$. The starting point $x_0 \in C$ and the CBF ensure we never leave this set. Taken from [9].

of a K-class function (as $h(0)$ might not be 0), but this has shown to not be a problem during our experiments.

Considering that following the rules can be represented as a set of positions, we now have a way to model rules as mathematical constraints, where $h(x)$ will represent a single rule.

5.2 Designing a speed filter with control barrier functions

With the definition of CBFs, the last step is to integrate them as a proper guarding mechanism. To do so, [8] proposes a quadratic optimization problem which can be seen as changing the nominal speeds commands. The quadratic program used is given by (10), where u_{nom} is the original speed computed for a robot.

$$u^*(t) = \arg\min_{u \in R^2} || \ u(t) - u_{nom} \ ||^2 \qquad s.t \ \ (*) \tag{10}$$

Constraints used for this optimization program are formulated using the set forward-invariance property defined earlier in (9). It is possible to have multiple constraints, where one constraint (or more) with a given $h_i(x)$ control barrier function will enforce the robot to respect a single rule. This optimization problem outputs an optimal control speed $u^*(t)$ with a speed close to the nominal one, which will be the transmitted command to the robots. Our objective is to enable or disable some of these constraints depending on the current state of the match.

The decision system being implemented using the Rust programming language [7], we used the *quadprog* library to implement it [12]. Note that there is always a solution to this quadratic problem, where $u(t) = 0$, since the current location $x(t)$ is inside the safe set C. The next goal is to benchmark the computation speed of this problem for each robot. If it is deemed that solving takes too much time for all robots, a possible solution is to deport this computation

inside the robots. Usage of CBFs for distributed collision avoidance has already been proven by [8].

5.3 Rules as mathematical constraints

With the defined speed filter, we may now incorporate respect of rules as constraints of the program. [9] gives a simple distance function to model virtual circular obstacles for the robots to avoid, given by (11), where D is the radius of the circular obstacle.

$$h(x) = \frac{1}{2}(||x - p_o||^2 - D^2) \tag{11}$$

We showcase the results of using such a function to approximate the defense area of the field by two separate virtual circular obstacles inside our qualification video [15], the next step being the modeling of all of the rules, so that new teams may use this method, lowering the barrier of entry. Ability to ensure safe control does not remove the need for path planning, as the target location might not be part of the safe set of positions the robot may attain. This filter is designed for behaviors that do not require achievement in some cases, to simplify the synthesis of strategies. For complex behaviors, this method may not be suitable, or should at least propagate the fact that the nominal speed has been modified to the strategy computations.

6 Acknowledgments

We would like to thank the University of Bordeaux for helping us during the creation of our association and the Bordeaux Institute of Technology for their continued support. Our sincere thanks to the previous team members for their prior work and to the CATIE who continue to help us improving the electronics. We want also to thank Tatsuya Ibuki for his work on control barrier functions and the help he provided to our team to achieve these results. Finally, thank you to Patrick Felix and Etienne Schimitz for the work they have done in the past that made it possible for the team to go this far.

Special thanks to Antoine Villoteau, Roland Kia, Victor Lucas Rosada Canesin, Mathieu François and Damien Delpy, the team of ENSEIRB-MATMECA students that accepted to work on the integration project of the Trinamic motor board for our team. Also, thank you to Grégoire Passault for accepting to oversee the project.

References

1. Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications, 2019.

2. A. Calugi, B.Chew, P. Félix, J. Gautier, C.Labbé C. Godinat, J. Lindois, T.W. Menier, and C.A. Vlamynck. Namec - team description paper small size league robocup 2025 application of qualification in division b, 2025.
3. V. L. Rosada Canesin, M. François, D. Delpy, R. Kia, A. Villoteau, and T. W. Menier. TMC4671 motor card end-of-year project, `https://github.com/Wanchai290/enseirb-pfa-trinamic`.
4. Ansys CES Edupack software, `https://www.ansys.com/fr-fr/products/materials/granta-edupack`.
5. Ames Aaron D., Grizzle Jessy W., and Tabuada Paulo. Control barrier function based quadratic programs with application to adaptive cruise control. *53rd IEEE Conference on Decision and Control*, 2014.
6. J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, Sep. 2004.
7. P. Félix, O. Ly, G. Passault, E. Schmitz, S. Loty, C. Laigle, A. Chauvel, T.W. Menier, V. Chaud, L. Paille, E. Miqueue, and B. Chew. Namec - team description paper small size league robocup 2023 application of qualification in division b, 2023.
8. Tatsuya Ibuki, Taichi Hirano, Riku Funada, and Mitsuji Sampei. Optimization-based distributed safety control with applications to collision avoidance for mobile robotic networks. *Advanced Robotics*, 37(1-2):87–98, 2023.
9. Thomas Wanchai MENIER. Distributed consensus for quadcopter formation control, 2025. `https://github.com/Wanchai290/consensus-control/blob/v1.0.0/reports/Report%20-%20Distributed%20consensus%20for%20quadcopter%20formation%20control%20-%20Thomas%20Wanchai%20MENIER.pdf`.
10. Masaaki Nagahara, Shun-Ichi Azuma, and Hyo-Sung Ahn. *Consensus Control*, pages 65–84. Springer International Publishing, Cham, 2024.
11. R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. In *Proceedings of the IEEE, vol. 95, no. 1, pp. 215-233*, Jan. 2007.
12. Crate quadprog documentation : `https://docs.rs/quadprog/latest/quadprog/`.
13. Nikhil P Raut and A.B. Kolekar. Experimental analysis of 3d printed specimens with different printing parameters for izod impact strength. *Materials Today: Proceedings*, 80:156–162, 2023. 3rd International Congress on Mechanical and Systems Engineering (CAMSE 2022).
14. TDP Search tool, hosted at `https://tdpsearch.com` as of time of writing. GitHub repository : `https://github.com/emielsteerneman/TDP`.
15. NAMeC 2026 SSL Qualification video `https://youtu.be/WNj9TYCHLfE`.
16. Xiangru Xu, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61, 2015.