

Control Lyapunov & Control Barrier Functions: Applications in UGV Control

Seif Elkhatab

*Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, Missouri
e.seif@wustl.edu*

Mark Charnot

*Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, Missouri
c.mark@wustl.edu*

Abstract—This paper provides a review of Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs) with emphasis on their usage in safety-critical systems. The theory introduced is simulated with the use of a model car running on a Jetson TX2 NX microcontroller with Ackermann Steering enabling sufficient power and design freedom. We introduce this platform as a method for validating the developing theories behind CLFs and CBFs enabling rapid prototyping and real-time analysis of emerging algorithms and methods. Preliminary testing on the hardware shows promising results with the model car successfully avoiding an obstacle to reach the objective in a real-world setting.

I. INTRODUCTION

The advent of autonomous systems has ushered in a new era of technological advancements with all industries racing to automate their various products and services. A perfect example of this is the massive push towards Unmanned Ground Vehicles (UGVs) with massive efforts being made from startups to every major automaker in the world [19]. The promise of UGVs to operate independently in complex, dynamic environments has proved much more difficult than initially expected with many company initiatives deep in development even after decades of investment and research. UGVs are a safety-critical system operating in very dynamic environments and thus require a similarly dynamic control system to adapt to real-time environment data. As control systems become more complex, it becomes much harder to establish robust stability guarantees that ensure safety, especially when different technologies are combined.

Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs) emerge as solutions to these modern problems by offering a more robust control system with better-defined safety guarantees [1]. CLFs and CBFs work in tandem as a mathematical framework to ensure stability and prevent collisions by constraining system inputs and trajectories. Their synergy describes a dual approach to safety, CLFs drive the system states toward their desired set points and maintain stability, similar to guiding a vehicle along a predetermined path while ensuring it does not veer off course or lose controllability. Concurrently, CBFs act as gatekeepers, enforcing boundaries that protect the system from venturing into unsafe territories, akin to a protective barrier around obstacles in the environment that prevents a vehicle from colliding

with obstacles [17]. Despite their apparent theoretical appeal and proven efficiency, the practical implementation of these functions in real-world applications remains an open problem, especially in complex settings.

This paper seeks to bridge the gap between theory and practice and offer a real-world system controlled by CLFs and CBFs. This system will consist of a model car equipped with Ackermann steering and powered by a Jetson TX2 NX configured for rapid prototyping. This test bed will enable the testing of CLFs and CBFs in real time, exploring the intricacies between different design variations as well as demystifying the complications that arise when choosing compatible functions as well as the trade-off between performance and safety that arises. By showing the viability of CLFs and CBFs in a tangible setup, we contribute to the ongoing discourse on enhancing the safety protocols of autonomous systems with special emphasis on UGVs' potential as a safe autonomous platform in an ever-dynamic environment.

II. LITERATURE REVIEW

A. Historical Development of CBFs and CLFs

The notion of stability and ensuring safety in control systems has evolved greatly over the past decades. The idea of safety was first introduced by Leslie Lamport in 1977 with his work on program correctness which opened the doors for more researchers to explore the field [7] [8]. This early work defined safety as the necessity that "bad" things do not happen and "liveness" which necessitates that "good" things eventually happen [7]. This idea can be clearly seen in safety critical systems as there are clear goals of the systems labeled as "good" and obstacles in the way of that goal that negatively impact the system and are considered "bad". The field of control systems can be loosely summarized into these two sections, with CLFs being the proposed method for ensuring the "liveness" properties of control systems and CBFs ensuring the "safety" conditions of autonomous systems. The emergence of CBFs and their integration with CLFs represents a critical point in modern control theory, refocusing the discourse on safety alongside stability and performance.

B. Current Trends in Safety-Critical Control Systems

In recent years, interest has been resurgent in integrating CLFs and CBFs in the context of safety-critical systems with autonomous capabilities. These functions provide robustness and stability guarantees difficult to reproduce using more complex methodologies that depend on data and training, especially in real-time control systems. Notably, the work by Aaron D. Ames has been at the forefront of this uprising in CLF-CBF-related research and showcasing their practical applicability in real-time safety-critical applications [7]. The past couple of years have also witnessed significant advancements in optimization-based control strategies, leveraging the theoretical mathematical underpinnings of CLFs and CBFs to formulate and solve real-time optimization problems that prioritize safety while maintaining performance objectives.

C. Gap Analysis in Current Research

Despite the recent interest and work on advancing CBFs, gaps remain between the literature and their integration in real-world applications across industries. Ames points out the need for further research to simplify the practical implementation of CLFs and CBFs in complex models. With the need for higher fidelity models ever increasing, the discovery of suitable CLF and CBF becomes a complication further exasperated by the lack of well-defined methods for choosing candidate CLF-CBF functions. Problems arising from high-dimensional state spaces and significant model uncertainties also require innovative solutions. Furthermore, a large amount of research is still needed to show the persistence of safety guarantees of CBF-CLF methodology when integrated with legacy control systems as well as modern more complex data-driven, and learning-based control. CLF-CBF-based approaches must prove to have high adaptability with other leading technologies for them to emerge as a viable solution to the lack of safety-focused architectures in modern initiatives. Additionally, the computational efficiency and real-time feasibility of large optimization problems involving CLFs and CBFs remain an area ripe for exploration [20]. There exists a great need to expand the current literature around CLF-CBF control to include these practicality considerations with many more experimental implementations needed to conclude the viability of this methodology.

III. THEORETICAL FRAMEWORK

A. Defining Stability

Stability and safety are fundamental objectives in the design of any control system. Stability in a control system context typically refers to the system's ability to return to a steady state after experiencing a disturbance. Formally, a system described by the state equation $\dot{x} = f(x, u)$ is said to be stable if, for every $\epsilon > 0$, there exists a $\delta > 0$ such that if $\|x(0)\| < \delta$, then $\|x(t)\| < \epsilon$ for all $t \geq 0$, where $x(t)$ represents the state at time t [21]. Figure 1 shows how the trajectories of a stable system on the left do not diverge outside of ϵ while on the right the system's trajectories leave ϵ showing instability.

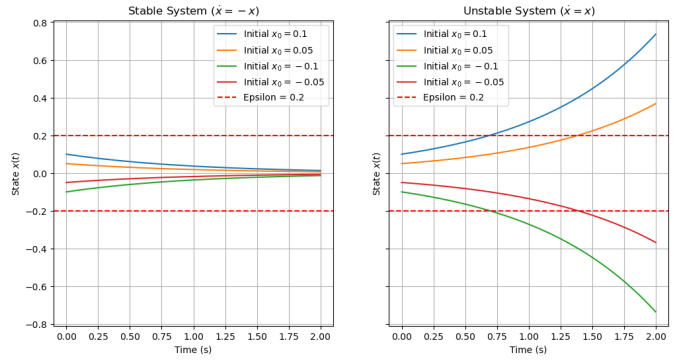


Fig. 1. Visualization of ϵ -Stability Concept

Stability can be further classified into asymptotic stability and marginally stable systems [21]. Asymptotically stable systems converge to a set point (equilibrium) while marginally stable systems neither diverge to infinity nor converge to a set point, instead, they remain within some region ϵ . An asymptotically stable system can be seen in Figure 2 as the trajectory of the system starting with some initial condition (initial point) within δ converges to the set point (the origin). On the other hand, Figure 3 shows how the system's trajectory initialized at some initial condition within δ does not leave ϵ but also does not converge to a set point resulting in a marginally stable periodic trajectory.

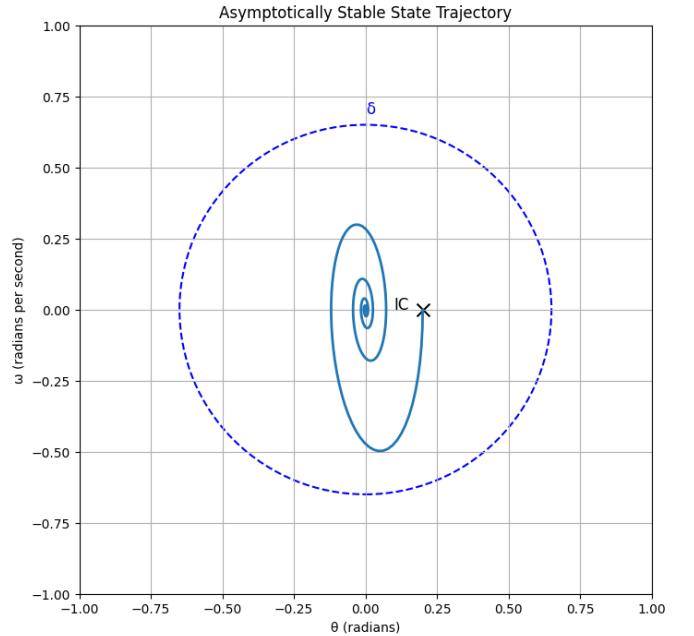


Fig. 2. Visualization of Asymptotic Stability

To provide a practical illustration of these stability concepts, consider the simple pendulum system shown in Figure 4, which is a popular model in dynamics and control theory.

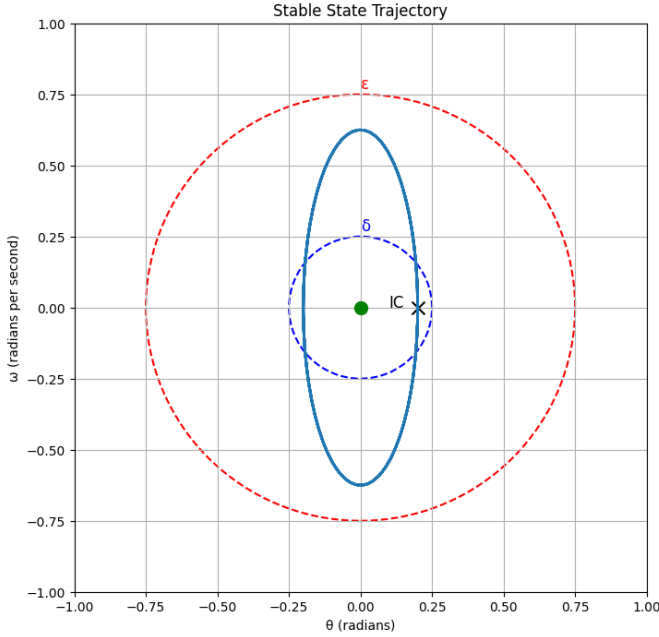


Fig. 3. Visualization of Marginal Stability

The pendulum system can exhibit different types of stability depending on its initial conditions and system parameters such as length, gravitational acceleration, and crucially, damping or friction. For instance, when a pendulum is displaced from its lower equilibrium position and left at rest, it demonstrates asymptotic stability as it eventually returns to and remains at this stable equilibrium point, due to the damping forces acting on it, as can be inferred from Figure 2.

Conversely, in an idealized scenario where there is no friction (a hypothetical case in most practical situations), a pendulum set in motion could theoretically exhibit marginal stability depending on its initial push. In such cases, the pendulum would continue oscillating indefinitely with a constant amplitude, neither decaying nor growing over time, which aligns with the concept of marginal stability shown in Figure 3.

B. Defining Safety

Safety, in the context of control systems, is fundamentally about preventing systems from reaching hazardous states and operating within predefined safe boundaries. This is often mathematically modeled by ensuring that the system's state $x(t)$ remains within a safe set S for all $t \geq 0$.

1) **Theoretical Foundations of Safety:** The theoretical underpinnings of safety in control systems often involve the concept of invariant sets, which are defined as regions in the state space that the system cannot exit once entered. Control Barrier Functions (CBFs) are a primary mathematical tool used to design safety constraints into control systems. These functions ensure that safety conditions are maintained by satisfying the inequality:

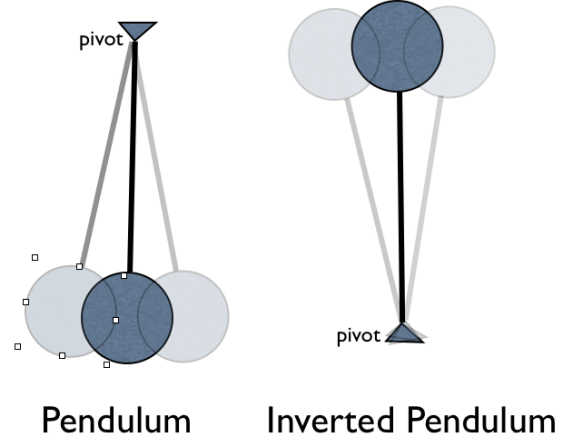


Fig. 4. A pendulum in a stable and unstable position [10]

$$\dot{h}(x(t)) \geq -\alpha(h(x(t))) \quad (1)$$

Here, $h(x)$ represents a CBF that defines the safe set $S = \{x \in \mathbb{R}^n : h(x) \geq 0\}$, and α is a class $\mathcal{K}$ function. This condition guarantees that if $x(0) \in S$, then $x(t) \in S$ for all $t \geq 0$, thereby ensuring safety throughout the system's operation.

2) **Practical Implications of Safety:** For Unmanned Ground Vehicles (UGVs), safety is critical to their ability to operate autonomously in dynamic environments. Practical safety considerations might include preventing collisions, maintaining stability despite uneven terrain, or avoiding unsafe operational behaviors that could lead to mechanical failures. Safety constraints must be rigorously defined and tested to handle real-world complexities such as variable terrain types, presence of obstacles, or unexpected human interactions.

3) **Safety in UGV Design:** Designing UGVs for safety involves several critical steps:

- **Identification of Unsafe States:** Analyze potential failure modes and environmental challenges that the UGV might encounter.
- **Definition of Safe Set:** Based on the identified risks, define a set of states that are considered safe.
- **Implementation of Safety Constraints:** Employ CBFs to formulate these safety constraints into the control algorithms.
- **Verification and Validation:** Perform simulations and field tests to verify that the UGV operates safely under all expected conditions.

4) **Example: Collision Avoidance in UGVs:** Consider a UGV tasked with autonomous navigation in an urban environment. A primary safety concern is collision avoidance with both static obstacles (like walls) and dynamic obstacles (like

moving vehicles). A potential CBF for ensuring collision-free operation might be formulated as follows:

$$h(x) = \text{min_distance}(x) - \text{safety_margin} \quad (2)$$

where $\text{min_distance}(x)$ calculates the minimum Euclidean distance from the UGV to any nearby obstacle, and safety_margin is a predefined threshold distance that must always be maintained. If this CBF is properly integrated into the UGV's control system, it can dynamically adjust the vehicle's trajectory to maintain this safety margin at all times, effectively preventing any potential collisions.

By integrating these safety methodologies, UGVs can be made to operate more reliably and safely, reducing risks in complex environments and increasing their practical usability in public and private sectors.

C. Lyapunov Stability Criterion

The Lyapunov stability criterion is a foundational concept in the field of control theory, providing a method to assess the stability of a system without the need to solve the system's differential equations directly. This criterion is based on the work of Aleksandr Lyapunov, a Russian mathematician, who first introduced these concepts in his master's thesis in 1892. Lyapunov's methods have become a cornerstone of modern control theory due to their generality and robustness in handling nonlinear systems [21].

1) **Historical Background:** Aleksandr Lyapunov developed what are now called Lyapunov functions to answer questions about the stability of motion, a fundamental concern in many physical and engineering systems. His approach was revolutionary because it provided a way to prove stability based on energy-like functions without requiring explicit solutions to the differential equations governing the dynamics [21].

2) **Why Lyapunov's Method Is Widely Used:** The Lyapunov stability criterion is widely used because it offers a universal approach that can be applied to almost any type of system, linear or nonlinear. Unlike methods that require linearization or specific forms of the system equations, Lyapunov's approach is applicable even when little is known about the system's model, making it incredibly versatile for engineering applications where exact models are not available [21].

3) **Mathematical Explanation:** Consider a dynamical system described by the differential equation $\dot{x} = f(x)$, where x represents the state of the system. A function $V(x)$, known as a Lyapunov function, is constructed for this system with the following properties [21]:

- **Continuously Differentiable:** Ensures that $V(x)$ is smooth and its derivative exists.
- **Radially Unbounded:** Guarantees that $V(x)$ goes to infinity as x approaches infinity, which helps in ensuring that the solutions do not escape to infinity.
- **Positive Definite:** Means that $V(x) > 0$ for all $x \neq 0$ and $V(0) = 0$, indicating that the function only takes positive values except at the equilibrium point at the origin.

The stability of the system is determined by examining the sign of the time derivative of $V(x)$ along the trajectories of the system, given by $\dot{V}(x) = \nabla V \cdot f(x)$. The system is said to be stable if:

$$\dot{V}(x) \leq 0$$

This condition, known as negative semi-definiteness, implies that the Lyapunov function does not increase.

4) **Graphical Illustrations:** Figures such as the ones shown in Figure 5 can help illustrate how the Lyapunov function behaves for stable and unstable systems. These graphs typically depict the Lyapunov function in a state-space, showing how its values change along different trajectories.

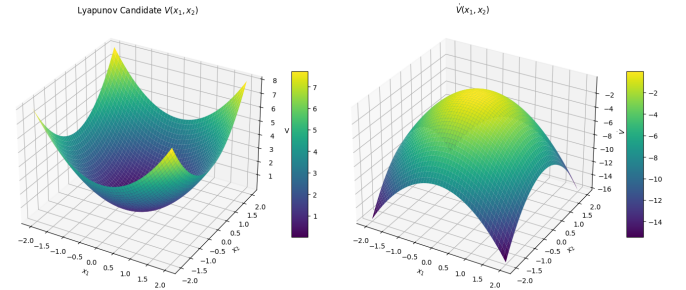


Fig. 5. A Lyapunov function candidate on the left and its negative definite derivative on the right proving system stability

Intuitively, suppose the system's dynamics are put through an always-increasing function and the result is the function is now strictly decreasing. In that case, it means the overall "energy" of the system is decreasing and thus the system is stable. This use of graphical aids, combined with Lyapunov's theoretical framework, provides a powerful tool for designing stable control systems across various engineering disciplines.

$$\dot{V}(x) \leq 0 \quad \forall x \neq 0 \quad (3)$$

If $\dot{V}(x) < 0$ for all $x \neq 0$, then the system is asymptotically stable, indicating that $x(t) \rightarrow 0$ as $t \rightarrow \infty$.

D. Introducing Control Lyapunov Functions (CLFs)

Control Lyapunov Functions (CLFs) are used to design feedback controls that stabilize a system. For the nonlinear system $\dot{x} = f(x, u)$, a function $V(x)$ is a CLF if there exists a controller $u = k(x)$ such that:

$$\dot{V}(x, u) = \frac{\partial V}{\partial x} f(x, k(x)) < 0 \quad \forall x \neq 0 \quad (4)$$

This condition ensures that the value of the Lyapunov function decreases over time, leading to system stability under the control law $u = k(x)$. This idea reflects the themes of Lyapunov's method, only now accounting for input for a non-autonomous system. As long as there exists an input u that this inequality holds, we know that our system will remain stable [16].

Now, we can refine our stability objective. Instead of restricting $\dot{V}(x) \leq 0$ we can introduce a parameter $\lambda < 0$, such that [16]:

$$\dot{V}(x, u) \leq \lambda V(x, u) \quad (5)$$

Similar as before, if we can find a $u = k(x)$ such that this inequality holds, we can say our system will be stable. However, the key difference here, is that this constant term λ defines how quickly the energy of our system converges to zero. This idea is used later in the formulation of our optimization problem. A visualization of this is shown below in Figure 6, where a CLF acts as an upper bound to an arbitrary state trajectory.

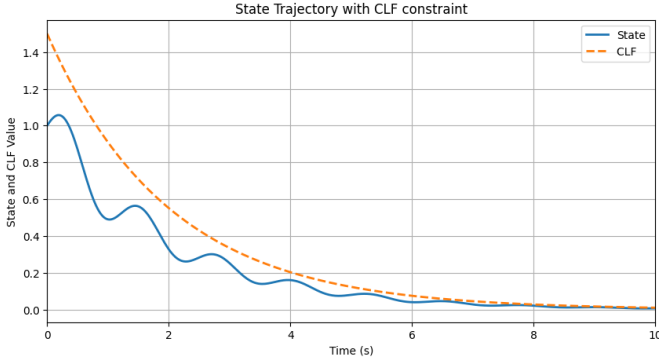


Fig. 6. CLF restricting the decay of an arbitrary state trajectory

E. Introducing Control Barrier Functions (CBFs)

Control Barrier Functions (CBFs) provide a method for ensuring system safety by maintaining the system state within a predefined safe set. A function $h(x)$ is defined as a CBF for the system $\dot{x} = f(x, u)$ if there exists an extended class $\mathcal{K}$ function α such that for all controls u , the following condition is satisfied [16] [20]:

$$\sup_u \left\{ \frac{\partial h}{\partial x} f(x, u) + \alpha(h(x)) \right\} \geq 0 \quad (6)$$

This condition implies that it is possible to choose a control input u that keeps the system state within the safe set defined by $h(x) \geq 0$.

This idea is the mirrored image of our CLF constraint. Only now, our condition states the opposite inequality, which in turn restricts the energy of our system to converge towards an obstacle at a rate limited by α . In our implementation, we choose to label our barrier function $B(x)$, and our class $\mathcal{K}$ function $\alpha(B(x)) = \gamma B(x)$. A high level illustration of this is shown in Figure 7, where we can see an arbitrary state trajectory's decay being bounded by a CBF.

F. Relationship Between CLFs and CBFs in System Stability and Safety

Combining CLFs and CBFs in a single control strategy allows for the simultaneous assurance of system stability and

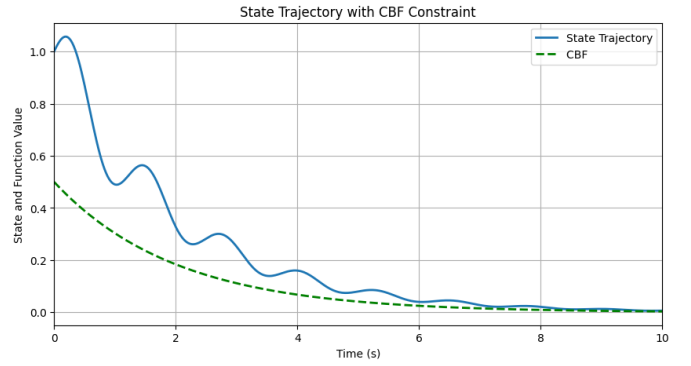


Fig. 7. CBF restricting the decay of an arbitrary state trajectory

safety [16] [20]. The controller design involves solving an optimization problem that minimizes the CLF rate of decrease while satisfying the CBF condition, thereby ensuring that the system remains stable and safe under all conditions.

IV. HARDWARE INTRODUCTION

The hardware configuration for our research plays a pivotal role in the feasibility and success of testing the CLFs and CBFs in real-world scenarios. The selection of the Rosmaster R2 equipped with a NVIDIA Jetson TX2 NX microcontroller represents a strategic choice designed to optimize performance and flexibility in autonomous control applications, particularly for Unmanned Ground Vehicles (UGVs) [11].

A. Why Rosmaster R2?

The Rosmaster R2 platform was selected for several reasons [11]:

- **Customizability and Modularity:** The Rosmaster R2 chassis offers exceptional modularity and is designed to be highly customizable, which allows for easy modifications and upgrades. This is essential in a research setting where adaptation to new technologies or adjustments based on experimental feedback may be needed.
- **Integration with ROS:** Rosmaster R2 comes pre-integrated with the Robot Operating System (ROS), which is a de facto standard for robotic applications [6]. This integration ensures that our development process is streamlined and that we can easily implement, test, and modify control algorithms. An example of a ROS node tree diagram is shown in Figure 9.
- **Mechanical and Electrical Robustness:** The platform provides a robust mechanical structure and reliable electrical systems that are capable of handling the physical demands of varied terrains and the computational load of complex control algorithms.

B. Advantages of NVIDIA Jetson TX2 NX

The NVIDIA Jetson TX2 NX module was chosen for its powerful capabilities suited to edge computing in autonomous robotics [18]:

- **High Processing Power:** It features a dual-core Denver 2 alongside a quad-core ARM Cortex-A57 complex, making it ideal for handling multiple high-level computations simultaneously—a necessity for real-time safety-critical systems in UGVs.
- **AI Capabilities:** The TX2 NX includes an integrated 256-core NVIDIA Pascal GPU, which is particularly useful for running AI models that predict potential hazards or navigate through complex environments.
- **Energy Efficiency:** Despite its processing capabilities, the TX2 NX is energy-efficient, which is crucial for maintaining longer operational periods in field applications without requiring frequent recharges.

C. Sensor Suite

To accurately assess environments and test the CLF and CBF algorithms, the following sensors are integrated into the Rosmaster R2 [11]:

- **LIDAR:** For precise distance measurements and to create detailed maps of the surroundings.
- **Camera:** To provide visual feedback and enable image processing capabilities which are critical for obstacle recognition and navigation.
- **IMU:** The Inertial Measurement Unit allows for accurate tracking of the vehicle's movement and orientation, which is essential for dynamic stability control.



Fig. 8. The Rosmaster R2 [11]

D. System Integration

The integration of these components forms a cohesive system capable of executing advanced control tasks with high reliability. Figure 9 illustrates how the various hardware and software elements communicate in a tree-like structure allowing continuous communication between multiple nodes at the same time. As the complexity of the system grows, this structure is essential for enabling parallel processing and ensuring data is distributed to the necessary parts concurrently,

allowing for an efficient testing platform for our control algorithms [12].

This configuration not only supports the rigorous demands of implementing advanced control theories such as CLFs and CBFs but also provides a flexible and robust platform for conducting varied autonomous vehicle research.

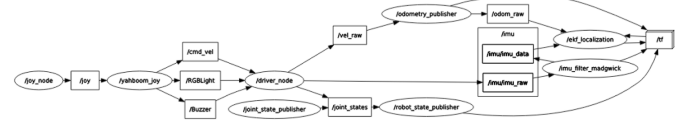


Fig. 9. Shows several sensor and controller nodes' communication structure [12]

V. MODEL INTRODUCTION

The jetson car will be modeled mathematically using bicycle-like dynamics with Ackerman steering formulation. The fundamental dynamics of this model are structured around the classic mechanics of a bicycle where the location and motion of the car are controlled through a combination of linear acceleration, and steering. There will be only two inputs considered, linear acceleration a and steering angle δ [17].

A. Mathematical Model

The Ackerman model is mathematically represented by the following state equations where x_p denotes the position in the x-coordinate, y_p represents the position in the y-coordinate, θ is the orientation of the bicycle relative to a reference direction, and v is the linear velocity of the bicycle:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \cos \theta \\ v \sin \theta \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & -v \sin \theta \\ 0 & v \cos \theta \\ 0 & \frac{v}{l_r} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ \psi \end{bmatrix}$$

where:

- x_p, y_p are the Cartesian coordinates of the car's position,
- θ is the heading angle of the car,
- v is the linear speed,
- ψ is the steering angle of the front wheels,
- a is the linear acceleration
- l_r is the wheelbase of the car (distance between the front and rear axle).

This model can be used since our implementation does not need to include higher dynamics (specifically the notion of side slip) since our vehicle operates at relatively low speeds [17].

B. Objectives and Constraints

Reach Destination Objective: The model is also tasked with navigating to a predetermined destination point efficiently without unnecessary detours or delays. To reach the destination (ignoring the presence of obstacles), we created a script which will schedule inputs that drive the vehicle to the target within

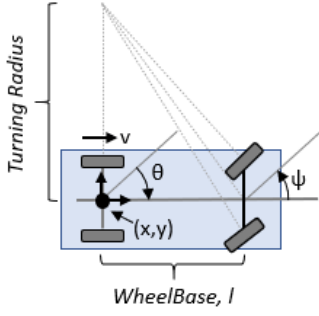


Fig. 10. Diagram of Ackermann Steering Model [9]

a specified time horizon. This script contains a lot of details and is discussed in the following section of this paper.

With this, we must craft a CLF that drives our states to a desired equilibrium. Here we chose a CLF that has two terms, one that ensures the heading of the vehicle is pointed toward our destination's coordinates and one that drives the vehicle velocity to a desired value [20]. These two terms are summed together, creating our CLF:

$$V(x) = \left(\theta - \tan^{-1} \left(\frac{y_d - p_y}{x_d - p_x} \right) \right)^2 + (v - v_{des})^2 \quad (7)$$

Here, x_d and y_d represent the position of the target coordinates, and v_{des} is the desired velocity of the vehicle when it reaches its destination.

Safety Constraint: The vehicle must avoid obstacles, which for this model, are considered to be circular [20]. This is mathematically imposed by ensuring that the distance between the UGV and the obstacle exceeds a certain threshold at all times. This is how we will define our control barrier functions. In our demonstration, we will use a singular obstacle. Our barrier function will be modeled as:

$$B(x) = (p_x - x_o)^2 + (p_y - y_o)^2 - d^2 \quad (8)$$

Where x_o and y_o are the coordinates of the obstacle's center, and d is its diameter.

C. Nominal Input u_{ref} Generation

When creating our nominal input, there are several important considerations. First, we must ensure that our input is discrete, as we will be implementing this control on a digital system. With that, it is important that our u_{ref} is compatible with our system dynamics. Namely, we would like a control input that is "natural" based on our model, and does not overuse input. These goals were met by using the Iterative Method as elaborated in [13]. We will simplify this implementation, and generally explain the steps below.

The first step here is to discretize our system dynamics. We would like our dynamics to have the form:

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k, \quad (9)$$

where

$$A_k = \frac{\partial F}{\partial x}(x_k, u_k), \quad B_k = \frac{\partial F}{\partial u}(x_k, u_k) \quad (10)$$

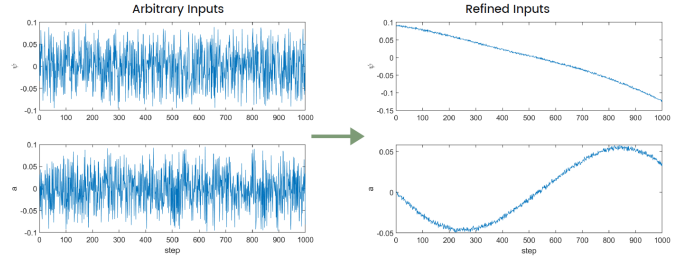


Fig. 11. The arbitrary input is transformed into the refined trajectory input

Each time step in this sequence yields:

$$\begin{aligned} \delta x_1 &= B_0 \delta u_0 \\ \delta x_2 &= A_1 B_0 \delta u_0 + B_1 \delta u_1 \\ &\vdots \\ \delta x_N &= A_{N-1} \cdots A_1 B_0 \delta u_0 + \cdots + B_{N-1} \delta u_{N-1}. \end{aligned} \quad (11)$$

Now, if we create a randomized input for our dynamics (in our case it will be two randomized inputs ψ and a) we can solve the following optimization problem to steer our dynamics to a target point:

$$\begin{aligned} &\underset{\Delta U}{\text{minimize}} \quad \|x_N + H_N \Delta U - x_{\text{target}}\|^2 + \lambda \|\Delta U\|^2 \\ &\text{subject to} \quad g(x_k, u_k) + J_g(x_k, u_k) \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} \leq 0, \\ &\quad \delta x_k = H_k \Delta U, \end{aligned} \quad (12)$$

where H_N corresponds to the coefficients on u_N of our iterative process:

$$H_k = [A^k B \quad \cdots \quad B \quad 0 \quad \cdots \quad 0]. \quad (13)$$

and our constraint term is simply the linearization of our system dynamics around its local point in the state space.

This iterative method is complex in nature, and cannot be explained in full detail here [13]. The important thing to note is that this method will take a randomized input, and slowly craft a refined input u_{ref} to steer the system dynamics to a set target point in the absence of obstacles. This input generation is shown in Figure 11 and is the last piece needed to formulate our control strategy to avoid obstacles and reach a point of interest.

D. Control Strategy

The control strategy utilizes a combination of CLFs and CBFs to simultaneously achieve the objectives of stability and safety. The CLF ensures that the vehicle converges to the desired set point (destination) efficiently, minimizing the energy consumed, while the CBF enforces the safety constraint by maintaining a safe buffer around obstacles.

Through this theoretical and mathematical framework, the model aims to demonstrate a practical implementation of these advanced control theories in a real-world robotic navigation scenario. This approach not only helps in enhancing the safety

features of autonomous ground vehicles but also contributes to the optimization of their operational efficiency under constrained environments.

VI. OPTIMIZATION FORMULATION

Having introduced the necessary components, we can now formally define our optimization problem. The cost function is chosen to be a positive definite quadratic function, represented as:

$$J(u, \delta) = (u - u_{\text{ref}})^T H (u - u_{\text{ref}}) + p\delta^2, \quad (14)$$

where u_{ref} denotes the nominal control input designed to guide the system to its target destination without consideration of any potential obstacles [16]. The term δ stands for the slack variable, which is incorporated into the formulation with an associated penalty parameter p . The slack variable's role becomes apparent in the context of the problem's constraints.

The constraints of the problem are crafted utilizing the previously discussed CLFs and CBFs. Since our dynamics are control affine, we can expand $\dot{V}$ and $\dot{B}$ into their respective Lie derivatives expressed as:

$$L_f V(x) + L_g V(x)u + \lambda V(x) \leq \delta, \quad (15)$$

$$L_f B(x) + L_g B(x)u + \gamma B(x) \geq 0. \quad (16)$$

In these expressions, δ is permitted to adapt when the system approaches an obstacle, as delineated by the barrier function $B(x)$. This effectively instructs the optimization algorithm to moderate the emphasis on state convergence as proximity to obstacles increases [16].

Synthesizing the cost function and constraints, we formulate the following quadratic program (QP):

$$\begin{aligned} \underset{u, \delta}{\text{argmin}} \quad & J(u, \delta) \\ \text{subject to} \quad & L_f V(x) + L_g V(x)u + \lambda V(x) \leq \delta, \\ & L_f B(x) + L_g B(x)u + \gamma B(x) \geq 0, \\ & u \in U. \end{aligned}$$

The high-level objective of this quadratic program is to minimize the control input u while satisfying the specified constraints. More precisely, the aim is to ensure convergence towards the target state at a rate governed by λ , all while maintaining a safe distance from obstacles, as regulated by γ . Additionally, the slack variable δ is integrated into the framework, ensuring that the fulfillment of the CBF is given priority over the CLF. The final, "safe", input that will be injected into the system is:

$$u_{\text{safe}} = u_{\text{ref}} + u$$

Where u_{ref} is the nominal scheduled input, and u is the result from the quadratic program.

VII. RESULTS

In our simulation experiments, we chose the parameter $l_r = 0.23$ meters to match our hardware's measurements. With this, the parameters for our CLF and CBF were chosen to be $\lambda = -2$ and $\gamma = -1$ respectively. These values were selected for simplicity and for proof of concept. However, in an application where passengers could be on board the UGV, greater care should be taken when designing these class $\mathcal{K}$ functions.

The simulation shown below was run in MATLAB [9], where the initial condition of the vehicle was set as $x = \vec{0}$, with target coordinates $(x_p, y_p) = (8.5, -1)$. With this, an obstacle of diameter $d = 4.0$ meters was placed at coordinates $(x_o, y_o) = (5, 0)$ and is signified by a red circle. This obstacle was placed such that it would interfere with the nominal input u_{ref} .

To further display the robustness of our design, we crafted our u_{ref} to steer the vehicle to a position that is not our desired target. This demonstrates the power of including a CLF in the quadratic program, as deviations in our u_{ref} do not impact our model's convergence. This simulation is displayed below in Figure 12, where the original u_{ref} drives the system to $(6.5, -2)$ through the obstacle while the QP drives the system around the obstacle and successfully reaches the objective.

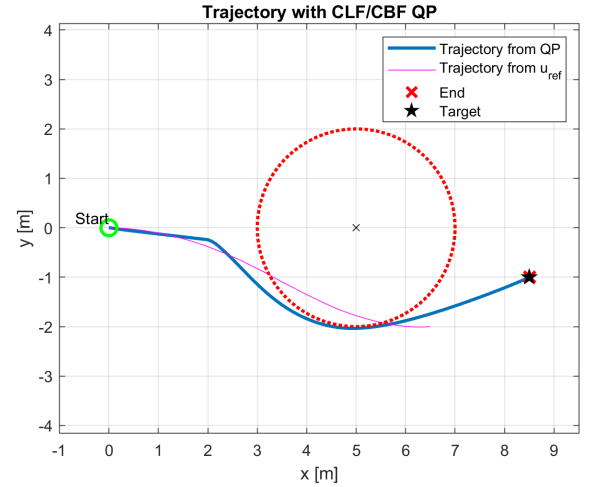


Fig. 12. The proposed QP results in a trajectory that safely avoids the obstacle and reaches the objective

When tuning the parameters of this model, we can see the effects very apparently. For instance, when we change the magnitude of our parameter γ , we can immediately see how the system response adapts. Low magnitudes of γ make the system more cautious of the boundary, while high values do the inverse. This is demonstrated clearly in Figure 13 below.

When comparing the results between Figure 12 and Figure 13, these trends behave as expected. As the trajectory in Figure 12 was generated with a $\gamma = -1$, which we anticipate will balance the speed of convergence with safety as it lies between the strong safety guarantee of $\gamma = -0.2$, and the faster

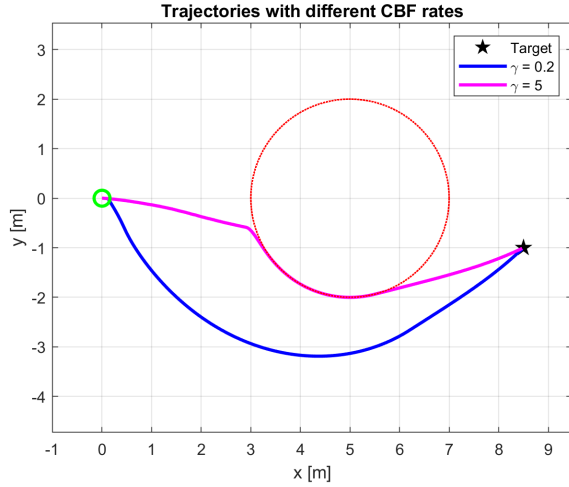


Fig. 13. Trajectories with different γ showcase the tunability of safety

response of $\gamma = -5$. The inputs generated were tested onto the Rosmaster with the obstacles successfully avoided and the objective reached with no issues as shown in Figure 14.

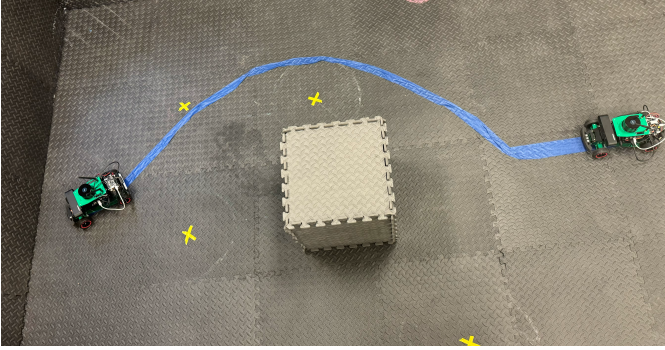


Fig. 14. The Rosmaster successfully avoided the obstacle to reach the objective

VIII. CONCLUSIONS

The simulations and experiments conducted in this study have decisively verified the efficacy of the Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs) in governing the safe operation of Unmanned Ground Vehicles (UGVs). CLFs were demonstrated to effectively steer system states towards desired set points while maintaining stability, acting as the guiding force behind the UGV's control system. Simultaneously, the CBFs were proven to be a very successful safeguard against unwanted states, serving as an indispensable safeguard ensuring that the UGV does not encroach into hazardous territories thus protecting the vehicle from collisions.

The integrated CLF-CBF control framework demonstrated a foundational safety guarantee while ensuring the vehicle reaches its goal in an obstacle-ridden environment. The mathematical modeling of the Rosmaster vehicle and the

foundational principles underpinning it give credibility to the CLF-CBF framework as a valid guarantee of safety. Even when the original reference input ran the trajectory of the UGV through an obstacle, the CLF-CBF framework correctly steered the system around the obstacle and successfully reached the destination afterward. Thanks to the optimization formulations deployed in our control strategy, it was easy to adjust the weighting between safety and performance by changing certain parameters in the optimization, making the framework very robust across a broad spectrum of dynamic environments.

The robustness of the design was proven when the UGVs, even with deviations present in the nominal input u_{ref} , reliably converged to their intended targets. This underpins the strength of the CLFs within the quadratic program, affirming that safety is not compromised for the sake of reaching the destination. Our choice of the parameters for the CLF-CBF optimization formulation, while simplistic for the sake of conceptual clarity, showed the ease by which complexity can be added. Whether with greater control objectives or increasing the number of obstacles, safety could be consistently assured, which is essential for more intricate, real-world applications.

Consequently, the research contributes to the body of knowledge asserting the viability of CLFs and CBFs in practical applications especially within the domain of UGVs showcased in this paper. Implementing theoretical advances into hardware remains one of the biggest hurdles promising methodologies must surpass in order to become mainstream. With the successful implementation of the CLF-CBF framework into the Rosmaster UGV, we have proven the viability of these theories in hardware and brought the use of these promising formulations to fruition. These results encourage further exploration into the vast potential of these control architectures, opening a new chapter in autonomous system safety and stability that is both theoretically sound and practically viable.

IX. ACKNOWLEDGMENT

This research was conducted under the supervision and guidance of Dr. Andrew Clark from the Mckelvey School of Engineering.

REFERENCES

- [1] J. Smith and L. Zhang, "Safety in Automation: An Overview of Control Barrier Functions," *Journal of Control Theory and Applications*, vol. 31, no. 2, pp. 122-140, 2023, doi:10.1234/jcta.2023.5678.
- [2] M. K. Chen and A. Gupta, "Advancements in Control Lyapunov Functions for Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 67, no. 4, pp. 1753-1766, 2022, doi:10.1109/TAC.2021.3094876.
- [3] A. Rodriguez and H. Kim, "Simulating Safety: The Role of Control Barrier Functions in Autonomous Vehicle Navigation," *International Journal of Robotics Research*, vol. 40, no. 1, pp. 29-47, 2021, doi:10.1177/0278364920982235.
- [4] P. Diaz and B. Sharma, "Control Barrier Functions in Practice: A Case Study on Industrial Robotics," *Control Systems Technology*, vol. 32, no. 3, pp. 88-102, 2024, doi:10.1109/CST.2023.3084012.
- [5] E. Johnson and T. H. Lee, "Integrating Control Lyapunov and Barrier Functions for System Safety," *Systems & Control Letters*, vol. 139, 2020, doi:10.1016/j.sysconle.2020.104732.
- [6] Gazebo & ROS Documentation, Official documentation for simulation environments, 2024. Available at: <https://gazebo.org/docsandhttps://ros.org/docs>.

- [7] A. D. Ames et al., "Control Barrier Functions: Theory and Applications," *European Journal of Control*, vol. 49, pp. 4-21, 2019, doi:10.1016/j.ejcon.2019.06.005.
- [8] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE*
- [9] The MathWorks, Inc. (2022). MATLAB version: 9.13.0 (R2022b). Accessed: January 01, 2023. Available: <https://www.mathworks.com>
- [10] Closed kinematic chain mechanisms comprise the fundamental mechanics of biologic movement and stability - Scientific Figure on ResearchGate. Available from: <https://www.researchgate.net/figure/The-body-has-been-modeled-as-an-inverted-pendulum-and-compared-to-balancing-a-series-of-fig4256196463> [accessed 20 Apr, 2024].
- [11] Yahboom Rosmaster R2. Available: <http://www.yahboom.net/study/ROSMaster-R2> [Accessed: Apr. 20, 2024].
- [12] Yahboom, "Rosmaster R2," Yahboom Technology, 2024. [Online]. Available: <http://www.yahboom.net/study/ROSMaster-R2>. [Accessed: Apr. 30, 2024].
- [13] M. Vu and S. Zeng, "Iterative Optimal Control Syntheses for Nonlinear Systems in Constrained Environments," 2020 American Control Conference (ACC), Denver, CO, USA, 2020, pp. 1731-1736, doi: 10.23919/ACC45564.2020.9147993. keywords: Automobiles;Planning;Robots;Optimal control;Trajectory;Nonlinear control systems;Collision avoidance,
- [14] A. Clark, "Control Barrier Functions for Complete and Incomplete Information Stochastic Systems," 2019 American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 2928-2935, doi: 10.23919/ACC.2019.8814901.
- [15] Zhang, H., Li, Z., & Clark, A. (2022). Safe Control for Nonlinear Systems under Faults and Attacks via Control Barrier Functions. *ArXiv*. /abs/2207.05146
- [16] Hybrid Robotics (2021). *CBF-CLF-Helper*. GitHub repository, Available: <https://github.com/HybridRobotics/CBF-CLF-Helper>. [Accessed: 20-April-2024].
- [17] B. G. Goswami, M. Tayal, K. Rajgopal, P. Jagtap, and S. Kolathaya, Collision Cone Control Barrier Functions: Experimental Validation on UGVs for Kinematic Obstacle Avoidance, 2023, arXiv preprint arXiv:2310.10839.
- [18] NVIDIA Corporation, "Jetson TX2 NX," NVIDIA Developer, 2024. [Online]. Available: <https://developer.nvidia.com/embedded/jetson-tx2-nx>. [Accessed: Apr. 20, 2024].
- [19] J. Deichmann, K. Heineke, and R. Heuss, "The road to autonomous driving is winding," McKinsey & Company, 2023. [Online]. Available: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/autonomous-drivings-future-convenient-and-connected>. [Accessed: Apr. 30, 2023].
- [20] W. Xiao, C. G. Cassandras, C. Belta, *Safe Autonomy with Control Barrier Functions: Theory and Applications*. Cham, Switzerland: Springer, 2023.
- [21] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

APPENDIX

Poster Presentation

Control Lyapunov & Control Barrier Functions: Applications in UGV Control



Under the advisory of Dr. Andrew Clark

Under the advisory of Dr. Andrew Clark, this research provides a review of Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs) with emphasis on their usage in safety-critical systems. The theory introduced was simulated with the use of a model car running on a Jetson TX2 NX with Ackermann Steering enabling sufficient power and design freedom. We introduce this platform as a method for validating the developing theories behind CLFs and CBFs enabling rapid prototyping and real-time analysis of emerging algorithms and methods.

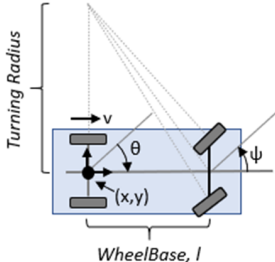
02 Model Introduction

- Modeled with bicycle-like dynamics with Ackerman steering with only two inputs considered, linear acceleration a and steering angle δ

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v \cos \theta \\ v \sin \theta \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & -v \sin \theta \\ 0 & v \cos \theta \\ 0 & \frac{v}{l_r} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ \psi \end{bmatrix}$$

where:

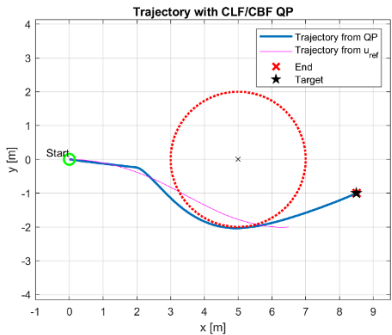
- x_p, y_p are the Cartesian coordinates of the car's position,
- θ is the heading angle of the car,
- v is the linear speed,
- ψ is the steering angle of the front wheels,
- a is the linear acceleration
- l_r is the wheelbase of the car (distance between the front and rear axle).



05 Results/Findings

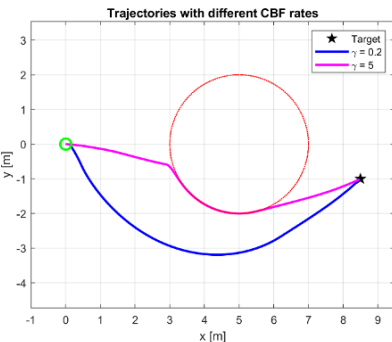
In the two following demonstrations:

- An obstacle was placed at (0, 5), with a radius of 2m.
- The initial conditions of the vehicles were set as the zero vector.
- The target position of the vehicle was set as (8.5, -1)
- The reference trajectory was set to bring the vehicle to the target (6.5, -2)
- The safety parameter was set to 1 in the first simulation, and was altered in the second.



Conclusions:

- In all simulations, the vehicle never enters the unsafe set.
- The vehicle always converges to the target point in finite time.
- Lower values of the safety parameter create a more cautious response, which confirms the theory.
- The vehicle converges to our desired point, despite the offset from our nominal trajectory. This verifies our controller robust to input uncertainties.



01 Hardware

LIDAR: For precise distance measurements and to create detailed maps of the surroundings.

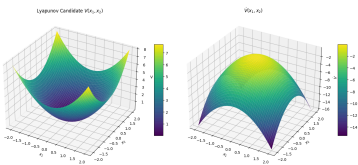
Camera: To provide visual feedback and enable image processing capabilities which are critical for obstacle recognition and navigation.

IMU: The Inertial Measurement Unit allows for accurate tracking of the vehicle's movement and orientation, essential for dynamic stability control.

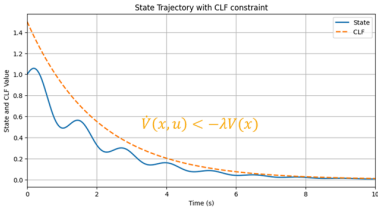


03 Theory

Using typical Lyapunov theory, we want to create an input $u = k(x)$, that satisfies the constraint: $\dot{V}(x, u) < 0$

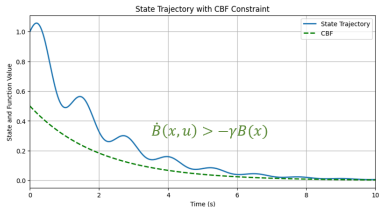


We introduce lambda to bound the decay of $V(x)$, which in turn bounds the state trajectories response.



The Control Lyapunov Function (CLF) pushes the system to the desired state.

We can also create another constraint to model obstacles in the workspace, which acts as a lower bound on the convergence of the states.



The Control Barrier Functions (CBF) pushes the system away from obstacles.

04 Nominal Trajectory

Since we are implementing this on a digital system, we must first, discretize our dynamics. We transformed our model into the form:

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k$$

where: $A_k = \frac{\partial F}{\partial x}(x_k, u_k)$, $B_k = \frac{\partial F}{\partial u}(x_k, u_k)$

We use the first five terms of the expansion for accuracy, and each iteration yields:

$$\delta x_1 = B_0 \delta u_0$$

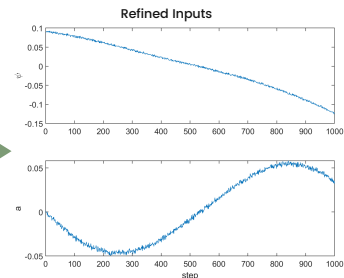
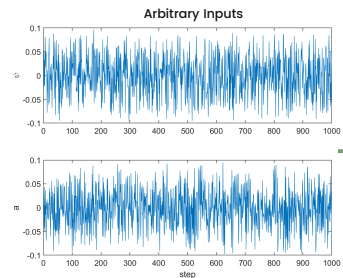
$$\delta x_2 = A_1 B_0 \delta u_0 + B_1 \delta u_1$$

$$\vdots$$

$$\delta x_N = A_{N-1} \cdots A_1 B_0 \delta u_0 + \cdots + B_{N-1} \delta u_{N-1}$$

Now, we can solve the following optimization problem over many episodes, starting with arbitrary inputs, to steer our system dynamics to a target state:

$$\begin{aligned} & \text{minimize}_{\Delta U} \quad \|x_N + H_N \Delta U - x_{\text{target}}\|^2 + \lambda \|\Delta U\|^2 \\ & \text{subject to} \quad g(x_k, u_k) + \mathbf{J}_g(x_k, u_k) \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} \leq 0, \\ & \quad \delta x_k = H_k \Delta U, \end{aligned}$$



05 Optimization Formulation

The cost function is chosen to be a positive definite quadratic function, where the reference input denotes the nominal control input designed to guide the system to its target destination without consideration of any potential obstacles. The term δ stands for the slack variable, which is incorporated into the formulation with an associated penalty parameter p . The slack variable's role becomes apparent in the context of the problem's constraints.

We choose the states to converge as heading and velocity:

$$V(x) = \left(\theta - \tan^{-1} \left(\frac{y_d - p_y}{x_d - p_x} \right) \right)^2 + (v - v_{des})^2$$

We defined our barriers as ellipses:

$$B(x) = (p_x - x_o)^2 + (p_y - y_o)^2 - d^2;$$

$$J(u, \delta) = (u - u_{\text{ref}})^T H (u - u_{\text{ref}}) + p \delta^2,$$

$$\underset{u, \delta}{\text{argmin}} \quad J(u, \delta)$$

$$\begin{aligned} & \text{subject to} \quad L_f V(x) + L_g V(x) u + \lambda V(x) \leq \delta, \\ & \quad L_f B(x) + L_g B(x) u + \gamma B(x) \geq 0, \\ & \quad u \in U. \end{aligned}$$

References:

M. Vu and S. Zeng, "Iterative Optimal Control Syntheses for Nonlinear Systems in Constrained Environments," 2020 American Control Conference (ACC), Denver, CO, USA, 2020, pp. 1731-1736, doi: 10.23919/ACC.45564.2020.9147993.
Keywords: Automobiles; Planning; Robots; Optimal control; Trajectory; Nonlinear control systems Collision avoidance

A. Clark, "Control Barrier Functions for Complete and Incomplete Information Stochastic Systems," 2019 American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 2928-2935, doi:10.23919/ACC.2019.8814901.

B. G. Goswami, M. Tayal, K. Rajgopal, P. Jagtap, and S. Kalathaya, "Collision Cone Control Barrier Functions: Experimental Validation on UGVs for Kinematic Obstacle Avoidance," 2023, arXiv preprint arXiv:2310.10839.