Case Study: Runtime Safety Verification of Neural Network Controlled System *

Frank Yang[1], Sinong Simon Zhan[1], Yixuan Wang[1], Chao Huang[2], and Qi Zhu[1]

[1] Electrical and Computer Engineering, Northwestern University, Evanston, USA
{frankyang2024,SinongZhan2028,yixuanwang2024,qzhu}@u.northwestern.edu
[2] School of Electronics and Computer Science, University of Southampton, Southampton, UK Chao.Huang@soton.ac.uk

Abstract. Neural networks are increasingly used in safety-critical applications such as robotics and autonomous vehicles. However, the deployment of neural-network-controlled systems (NNCSs) raises significant safety concerns. Many recent advances overlook critical aspects of verifying control and ensuring safety in real-time scenarios. This paper presents a case study on using POLAR-Express, a state-of-the-art NNCS reachability analysis tool, for runtime safety verification in a Turtlebot navigation system using LiDAR. The Turtlebot, equipped with a neural network controller for steering, operates in a complex environment with obstacles. We developed a safe online controller switching strategy that switches between the original NNCS controller and an obstacle avoidance controller based on the verification results. Our experiments, conducted in a ROS2 Flatland simulation environment, explore the capabilities and limitations of using POLAR-Express for runtime verification and demonstrate the effectiveness of our switching strategy.

1 Introduction

The increasing complexity of control strategies used in cyber-physical systems (CPSs) [34], specifically those based on neural networks, has revolutionized decision-making and control in several critical domains, including healthcare [58, 59], robotics [48, 49, 61], transportation [14, 39, 55], building control [53, 56, 57], and industrial automation [9, 54]. These advanced control approaches excel at handling complex and dynamic environments due to their ability to learn and adapt from data. However, assuring the safety and stability of these systems for the nonlinearity of control systems and their closed-loop formation with dynamic systems remains a significant challenge [4, 42, 64, 66].

Literature in this domain primarily focuses on developing methodologies to assess and guarantee the reliability and robustness of neural network decisions.

* Frank Yang, Simon Sinong Zhan, Yixuan Wang, and Qi Zhu's work is partially supported by US National Science Foundation grants 2324936 and 2328973. Chao Huang's work is supported by the grant EP/Y002644/1 under the EPSRC ECR International Collaboration Grants program, funded by the International Science Partnerships Fund (ISPF) and the UK Research and Innovation.

Early approaches often relied on static analysis techniques that scrutinized network structures and weights to predict behavior under various inputs [1, 26, 46]. Recent advancements have introduced more dynamic methods, such as formal verification and reachability analysis [2, 18, 40, 50], which offer more nuanced insights into network behaviors across potential operational scenarios. The Simplex-based Reachability Analysis [7, 16] guarantees system overall safety by integrating a verified safety controller and decision logic that switches between complex and safety controllers. While making significant contributions on real-time reachability, the use of these verification tools in realistic environments with machine-learning components remains largely unexplored. Our work uses POLAR-Express [50], a state-of-the-art verification tool to perform online reachability analysis for realistic robotic systems with neural network control and LiDAR sensing. We demonstrate the effectiveness of POLAR-Express for online reachability analysis by safely navigating robots in complex environments with obstacle constraints. The main contributions of this paper are as follows:

- We present a comprehensive study demonstrating the feasibility of performing runtime verification by POLAR-Express on Turtlebot for safe navigation.
- We provide a safe online controller switching strategy to avoid unknown obstacles based on the runtime verification result.

2 Related Work

Runtime Verification for Control. Runtime Verification (RV) plays a crucial role in the real-time operation of autonomous systems, such as autonomous vehicles [25], transportation networks [41] and medical devices [35]. In control theory, techniques such as adaptive control and robust control are employed to manage uncertainties and ensure stability in real-time scenarios [5, 63]. From the formal methods perspective, model checking, which utilizes temporal logic specifications like linear temporal logic (LTL) and signal temporal logic (STL), forms the backbone of verification processes on the system's trajectories [8, 15, 22, 30, 44, 60, 62]. Moreover, the integration of stochastic quantification tools with temporal logic through conformal prediction frameworks offers a formal statistical guarantee of system reliability under dynamic conditions [10, 37]. These developments have fostered innovative hybrid approaches that combine the strengths of control theory and formal methods to tackle complex verification challenges in real-time systems [3, 45].

NNCS Verification. The verification of neural networks has emerged as a critical research area [65]. Tools like Reluplex [31], Marabou [32], and Sherlock [17] employ techniques derived from formal methods to ensure that neural networks adhere to specified safety and performance criteria. Others methods includes optimization-based over-approximation [17, 46], and hybrid system approximation [29]. Alongside these, with the existing techniques for verifying dynamic systems [11, 13], a series of works have attained considerable maturity, providing formal analysis for neural network controlled systems (NNCSs) [2, 18, 19, 20,

24, 33, 38, 47, 50, 51, 52]. However, most of these approaches have not demonstrated the capability to verify NNCSs in a runtime environment. [28] presented the feasibility of Verisig [27, 29] using high-dimensional LiDAR measurements as the NNCS input, albeit in a simplistic setting for runtime requirements.

3 POLAR-Express Case Study

3.1 Preliminary

NNCS. We consider the explicit dynamics of an NNCS as $\dot{s} = f(s, a)$ where the state variable is $s \in \mathcal{S} \subseteq \mathbb{R}^n$, control input is $a \in \mathcal{A} \subseteq \mathbb{R}^m$, and the dynamic $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a Lipschitz continuous function, ensuring a unique solution of the ODE. Such a system can be controlled by a feedback NN controller κ_{nn}, at i-th $(i = 0, 1, \cdots)$ sampling period $i\delta$, κ_{nn} reads the system state $s_{i\delta}$, generates a control input $a = \kappa_{nn}(s_{i\delta})$, and the system evolves according to $\dot{s} = f(s, a)$ within the period of $[i\delta, (i+1)\delta]$. The *flowmap* function $\varphi(s_0, t) : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is to describe the solution of the NNCS, which maps the initial state s_0 to the system state $\varphi(s_0, t)$ at time t starting from s_0. We call a state s' *reachable* if there exist $s_0 \in S$ and $t \in \mathbb{R}_{\geq 0}$ with $s' = \varphi(s_0, t)$. A reachable set $\mathcal{S}_r^T$ is a collection of all reachable states within a time range $T = \mathbb{R}_{\geq 0}$ given an initial space $\mathcal{S}_0 = \{s_0\}$, i.e., $\mathcal{S}_r^T = \{\varphi(s_0, t), \mid s_0 \in \mathcal{S} \wedge t \in T\}$. Intuitively, once the reachable set $\mathcal{S}_r^T$ is non-overlapping with the unsafe sets $\mathcal{S}_u$, safety is guaranteed for such an NNCS throughout the time horizon T.

POLAR-Express. POLAR-Express [50] is a reachability analysis tool for NNCS based on polynomial arithmetic, developed upon POLAR [23]. It uses Bernstein polynomial interpolation to over-approximate the non-differentiable activation functions to enable layer-by-layer Taylor-Models (TMs) propagation for general feed-forward neural networks. The output over-approximation from the neural network is combined with Flow* [12] for next-step reachable set computation. This process repeats with the previous reachable set result as the input set for the next step and thus rolls out the overall reachable set step by step within the entire time horizon. Moreover, to tighten the over-approximation, POLAR-Express stores the TM intervals symbolically with their linear transformation matrix and only evaluates the remainder interval at the end. This approach is called symbolic remainder, which reduces the accumulation of over-approximation error in TM by avoiding the wrapping effect in linear mappings.

3.2 Task Specification

We control the Turtlebot 3 Burger (Details in Appendix A) in the ROS2 Flatland simulation to execute a left turn via an NN controller in a structured environment bounded by 5-meter walls (Fig. 1). The Turtlebot is equipped with LiDAR sensing capabilities, enabling it to localize and detect obstacles within its surroundings. While the NN controller computes

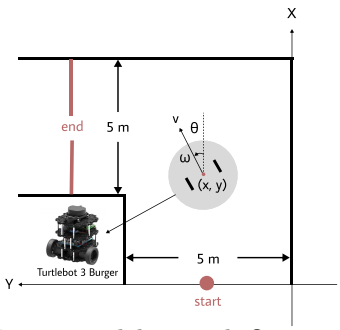


Fig. 1: Turtlebot Task Overview

the desired speed and steering angle for the left turn, POLAR-Express runs in real-time to verify the controller's safety. To create dynamic and uncertain environments, we introduced random obstacles during navigation, which do not exist in the training phase of the NN controller. This scenario ensured that some of the NN control signals would be unsafe, thereby requiring POLAR-Express to capture unsafe maneuvers in real time and demanding a safe control adaptation strategy.

3.3 Runtime Verification (POLAR-Express) based Safe Control

Fig. 2a outlines this case study's safe closed-loop control framework. We use POLAR-Express to compute reachable sets of NN controller κ_{nn} for Turtlebot at runtime. In case of a potential collision, the Turtlebot is switched to a backup obstacle avoidance controller κ_b for safety. We switch back to the NN controller if it is verified to be safe after the obstacle avoidance controller takes over. We introduce the details of each component in the following.

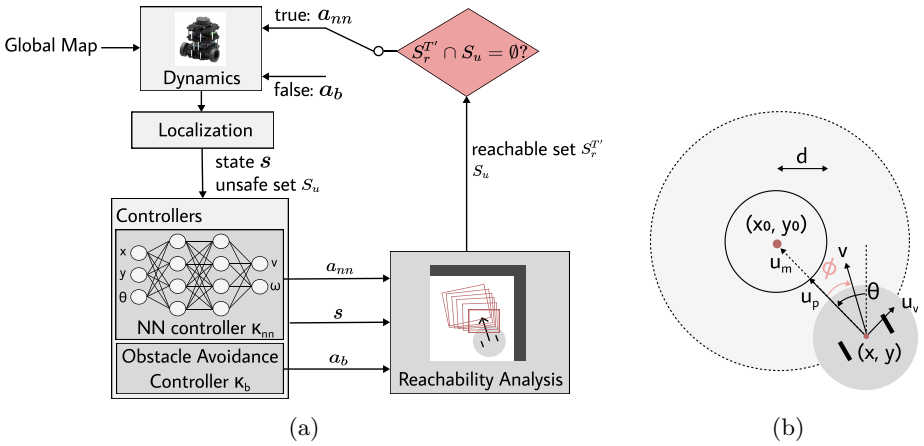


Fig. 2: **a**. The case study framework. We use POLAR-Express to determine the switch between an NN controller and an obstacle avoidance controller. **b**. The obstacle avoidance controller design moves the Turtlebot counterclockwise while keeping a safe distance.

Turtlebot Dynamics and localization. We model the Turtlebot's dynamics as $\dot{x} = \cos(\theta)v$, $\dot{y} = \sin(\theta)v$, $\dot{\theta} = \omega$ [43]. where θ is the orientation angle around x-axis and (x, y) is the localized position, $s = [x, y, \theta] \in \mathcal{S}$. $a = [v, \omega] \in \mathcal{A}$ is the control input signal, representing linear velocity and angular velocity, generated by the controller. Given a global map of the environment and the laser scan data from the LDS-01, the Turtlebot can localize its position by the Adaptive Monte-Carlo Localization (AMCL) approach implemented in the Nav2 package [6].

NN Controller κ_{nn}. We construct an NN controller $\kappa_{nn} : \mathbb{R}^3 \to \mathbb{R}^2$ with two hidden layers with a size of 64 neurons and ReLU activation functions. The controller takes the localized (x, y, θ) as input and outputs the linear velocity and angular velocity (v, ω) for the Turtlebot, i.e., $a_{nn} = (v, \omega) = \kappa_{nn}(x, y, \theta)$. To train the network, we collected 100 trajectories from expert demonstration data at 20 Hz in a simulation environment using the Nav2 goal package, moving the robot from a desired starting position to an end zone, and thus obtaining a dataset of $\{(x, y, \theta, v, \omega)\}$. We then train the NN controller via supervised learning to reduce an MSE loss as $\|\kappa_{nn}(x, y, \theta) - (v, \omega)\|^2$. It is worth noting that there are no obstacles in the environment during offline training.

Obstacle Avoidance Controller κ_b. If the runtime verification result of κ_{nn} is unsafe, we switch to the obstacle avoidance controller κ_b. Given the obstacle position, κ_b move the Turtlebot around the obstacle counterclockwise while keeping a constant distance d by adapting the algorithm in [36], as shown in Fig. 2b. Let (x, y) be the robot's localized position and (x_0, y_0) be the obstacle's center. The distance vector from the robot is $u_m = \begin{bmatrix} x_0 - x \\ y_0 - y \end{bmatrix}$. To maintain a safe distance d, we compute $u_p = u_m - \frac{u_m}{||u_m||} * d$, which points toward the obstacle if $\|u_m\| \geq d$, and vice versa. To move in parallel with the obstacle, we rotate u_p by 90 degrees: where R$=\begin{bmatrix} 0, -1 \\ 1, 0 \end{bmatrix}$. Combining both components, the desired motion and angle for safe obstacle avoidance is $u = u_p + u_v$ and $\phi = \arctan(u_x, u_y)$, where u_x and u_y are the projections of u onto x- and y-axes, respectively. Considering Turtlebot's physical limits (0.22m/s linear and 2.84 rad/s angular), we cap the desired steering velocity v while setting it to $\|u\|$. Similarly, we cap the desired steering angle ω while setting it to the difference between ϕ and current orientation θ. The control input of κ_b becomes

$$a_b = (v, \omega) = (\min(0.22, ||u||), \min(2.84, \phi - \theta))$$

It is important to note that κ_b is a fallback mechanism to steer the robot to safety when κ_{nn} is deemed unsafe by runtime verification. For the scope of this work, we assume that κ_b is guaranteed to safely navigate the robot around the obstacle to a stable point where κ_{nn} can resume control.

Obstacle Detection as Unsafe Regions $\mathcal{S}_u$. We introduce obstacles of random size and location on the NNCS trajectory for online navigation. Note that the neural network does not have any knowledge of the random obstacle on the map. Rather, these obstacles can be detected and localized by the sensing ability of the Turtlebot at runtime. The location of these obstacles is treated as the unsafe region $\mathcal{S}_u$ for the safety verification of the neural network controlled Turtlebot by POLAR-Express.

Controller Switching Logic. As mentioned, Turtlebot can detect and localize the obstacles' locations as unsafe regions $\mathcal{S}_u$. At runtime, we use POLAR-Express to compute an over-approximation of reachable set for κ_{nn} starting from the current state s within a time horizon T' as $\mathcal{S}_r^{T'}$. If $\mathcal{S}_u$ overlaps with $\mathcal{S}_r^{T'}$, this

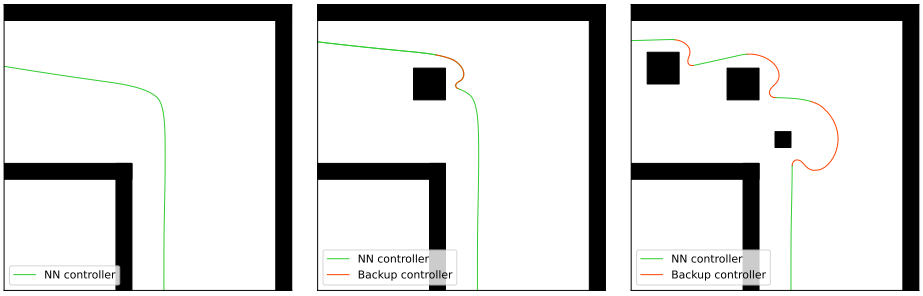
indicates a potential collision between Turtlebot under κ_{nn} and the obstacle within time horizon T', and therefore we switch to κ_b producing a_b for safety. While operating under κ_b, the robot continues to perform online reachability analysis for κ_{nn}. If $\mathcal{S}_u$ is no longer overlapping with $\mathcal{S}_r^{T'}$, i.e., $S_r^{T'} \cap S_u = \emptyset$, the robot switches back to κ_{nn}, as its control input is verified to be safe. This synergistic approach leverages κ_{nn} for efficient task execution and relies on κ_b and reachability analysis to guarantee safety in complex and cluttered environments; the switching logic can be carried entirely online.

4 Experiments

The simulation was performed on a Dell XPS 15 with an i7 processor, performing reachability analysis every 0.2 seconds in the callback function of the ROS2 Flatland simulation. POLAR-Express can be customized by adjusting key hyper-parameters such as the degree of the Taylor Model (TM), the order of the Bernstein Polynomial approximation (BP), and the number of verification steps (please see [11, 50] for more details of these hyper-parameters). By default, we assign the order of TM as 2 and the order of BP as 2 with 10 verification steps. With the default parameters, our framework operates effectively in both single (Fig. 3b) and multiple obstacle avoidance scenarios (Fig. 3c). Our well-trained κ_{nn} driving agent responds to obstacles detected with the reachable set computation by POLAR-Express timely and correctly activating the guarding condition, which then switches to the κ_b controller, also shown in Fig. 4. The κ_{nn} resumes control once the agent steers around the obstacle and the reachable set no longer overlaps with unsafe areas (Fig. 3b, Fig. 4). In the multiple-obstacles scenario, our runtime framework consistently manages several controller switches, ensuring safety throughout the operation (Fig. 3c). To comprehensively evaluate the case study, we then explore different parameter settings for POLAR-Express in different runtime scenarios, which may affect the tightness and computation efficiency of the reachable set.

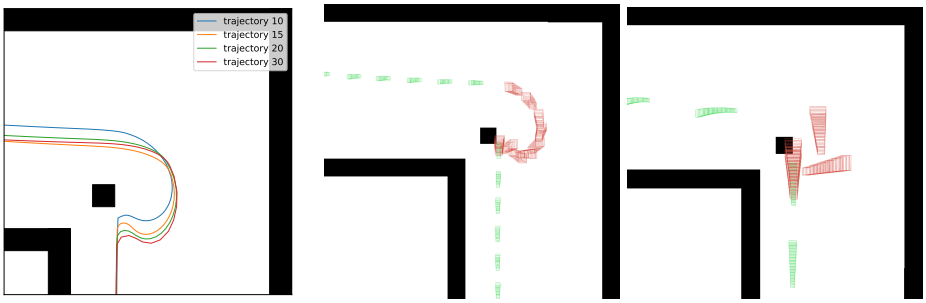
Verification Time Steps. The verification time step determines the temporal horizon over which POLAR-Express computes the reachable set of the robot's future states. As observed in Fig. 4a, longer verification time steps predict further and react to obstacles further ahead, while shorter steps react closer to obstacles. Although this predictive capability is desirable, increasing the verification time step introduces several drawbacks, as shown below.

Longer verification time steps increase the computational cost and may not satisfy the real-time verification requirement, as evidenced by the runtimes for different time steps in Table 1. Fig. 4 demonstrates that we continue using κ_{nn} if its runtime reachable set (green bounding boxes) does not overlap with the obstacle, and switch to κ_b otherwise, indicated by the red runtime reachable set. The visualization shows that controllers with longer verification time steps, such as 30 (Fig. 4c), compute less frequently than those with shorter time steps, like 10 (Fig. 4b). Secondly, longer verification time steps can cause the controller to



(a) No Obstacle (b) Single Obstacle (c) Multi-Obstacles

Fig. 3: The navigation trajectory of Turtlebot with our runtime verification based control by κ_{nn} (green) and κ_b (red) for **a).** No obstacles, **b).** navigating around a single obstacle, and **c).** navigating through multiple obstacles. The connection points of green and red are the controller switching points.



(a) trajectories (b) 10 time steps (c) 30 time steps

Fig. 4: Trajectories and runtime reachable set visualization by POLAR-Express with varying verification time steps: green boxes and red boxes show runtime reachable sets using the NN and the obstacle avoidance controller, respectively.

become more conservative and less task-critical, spending more time on obstacle avoidance and delaying task completion (Table 1). Lastly, longer verification steps may lead to an excessive accumulation of over-approximation error in the reachable set. This can result in an overly conservative evaluation of κ_{nn}'s safety, causing a premature switch to κ_b and consequent performance degradation.

Overall, trajectory planning systems face a trade-off between computational complexity and safety considerations. Longer verification time steps ensure safer navigation by exploring more potential paths and identifying obstacles earlier, but this comes at the cost of increased computational time and data sparsity, potentially causing delayed verification decisions and reduced task criticality. Conversely, shorter verification time steps may be computationally more efficient but risk overlooking potential obstacles or failing to plan adequately. Striking the right balance between these factors is crucial for performance and safety.

Verification Steps	10	15	20	30
Runtime (s)	0.18	0.26	0.35	0.53
Task Total Time Usage(s)	77.73	80.05	82.32	89.31
Obstacle Avoidance Controller Time Usage(s)	20.97	22.37	25.19	28.99
Obstacle Avoidance Controller Utilization (%)	26.97	27.94	30.6	32.46

Table 1: The verification time step vs. runtime (s)

Timing of Runtime Verification with POLAR-Express. In this section, we evaluate the runtime performance of POLAR-Express by conducting experiments with different combinations of Taylor Model (TM) degrees and Bernstein Polynomial (BP) approximation orders, which determines the accuracy of NN approximation and dynamic systems propagation. Intuitively, higher order degrees of the polynomials within POLAR-Express provide more powerful and accurate approximations but come with more computation burden.

The bar graph in Fig. 5 represents the runtime for various TM degrees and BP orders. Each runtime data is collected from the callback function and averaged for 10 trajectories. The POLAR-Express setup is fixed at 10 verification steps. In our evaluation, high TM degrees directly result in longer runtimes. We found that an increase in BP order does not drastically increase the time cost of the verification. Since the simulation is set to run the callback function every 0.2 seconds, only combinations with an average runtime of less than 0.2 seconds are considered valid for real-time performance. Based on this criterion, the valid combinations include TM degrees up to 3.

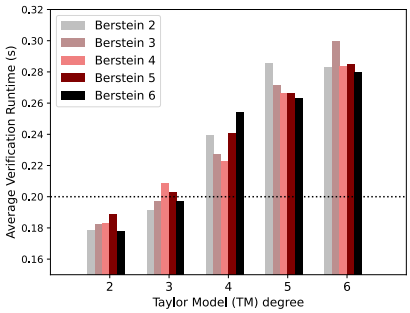


Fig. 5: The 10-step verification runtime of POLAR-Express with different TM and BP orders.

5 Conclusion and Future Work

This paper presents a runtime verification case study where an autonomous Turtlebot, equipped with a neural network (NN) controller, navigates a structured environment using only LiDAR measurements and POLAR-Express for runtime reachability analysis. Our research can expand in several directions: 1) Adapting our framework to accommodate system uncertainties and stochastic policies is a potential area for further development. 2) Incorporating scheduling techniques from the real-time systems for runtime verification could enhance system-level efficiency, where we can opportunistically call the verification engine only when it is necessary. 3) The switching logic of this case study is static and relatively simple, overlooking the fact that the obstacle avoidance controller could enter states that are not recoverable by the NN controllers. This could also be a future direction for improving this work.

Bibliography

[1] Albarghouthi, A., et al.: Introduction to neural network verification. Foundations and Trends® in Programming Languages **7**(1–2), 1–157 (2021)

[2] Althoff, M.: An introduction to cora 2015. In: Proc. of the workshop on applied verification for continuous and hybrid systems. pp. 120–151 (2015)

[3] Alur, R.: Principles of Cyber-Physical Systems. MIT Press (2015)

[4] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical computer science **138**(1), 3–34 (1995)

[5] Astrom, K.J., Murray, R.M.: Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press (2010)

[6] Author(s): Development of an automated benchmark for the analysis of nav2 controllers (Year), unpublished

[7] Bak, S., Johnson, T., Caccamo, M., Sha, L.: Real-time reachability for verified simplex design. ACM Transactions on Embedded Computing Systems **15**(26), 1–27 (2016)

[8] Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for ltl and tltl. ACM Transactions on Software Engineering and Methodology (TOSEM) **20**(4), 1–64 (2011)

[9] Breivold, H.P., Sandström, K.: Internet of things for industrial automation–challenges and technical solutions. In: 2015 IEEE International Conference on Data Science and Data Intensive Systems. pp. 532–539. IEEE (2015)

[10] Cairoli, F., Bortolussi, L., Paoletti, N.: Learning-based approaches to predictive monitoring with conformal statistical guarantees. In: International Conference on Runtime Verification. pp. 461–487. Springer (2023)

[11] Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for nonlinear hybrid systems. In: Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25. pp. 258–263. Springer (2013)

[12] Chen, X., Sankaranarayanan, S., Abrah´am, E.: Flow* 1.2: More effective to play with hybrid systems. Applied Verification for Continuous and Hybrid Systems pp. 152–159 (2015)

[13] Chutinan, A., Krogh, B.H.: Computational techniques for hybrid system verification. IEEE transactions on automatic control **48**(1), 64–75 (2003)

[14] Deka, L., Khan, S.M., Chowdhury, M., Ayres, N.: Transportation cyber-physical system and its importance for future mobility. In: Transportation cyber-physical systems, pp. 1–20. Elsevier (2018)

[15] Desai, A., Dreossi, T., Seshia, S.A.: Combining model checking and runtime verification for safe robotics. In: International Conference on Runtime Verification. pp. 172–189. Springer (2017)

[16] Desai, A., Ghosh, S., Seshia, S.A., Shankar, N., Tiwari, A.: Soter: A runtime assurance framework for programming safe robotics systems. In: 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems

and Networks (DSN). pp. 138–150 (2019). https://doi.org/10.1109/DSN.2019.00027

[17] Dutta, S., Chen, X., Jha, S., Sankaranarayanan, S., Tiwari, A.: Sherlock-a tool for verification of neural network feedback systems: demo abstract. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 262–263 (2019)

[18] Dutta, S., Chen, X., Sankaranarayanan, S.: Reachability analysis for neural feedback systems using regressive polynomial rule inference. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 157–168 (2019)

[19] Fan, J., Huang, C., Chen, X., Li, W., Zhu, Q.: Reachnn*: A tool for reachability analysis of neural-network controlled systems. In: International Symposium on Automated Technology for Verification and Analysis. pp. 537–542. Springer (2020)

[20] Fan, J., Huang, C., Li, W., Chen, X., Zhu, Q.: Towards verification-aware knowledge distillation for neural-network controlled systems: Invited paper. In: 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pp. 1–8 (2019). https://doi.org/10.1109/ICCAD45719.2019.8942059

[21] Gross, D.: An Implementation Approach of the Gap Navigation Tree Using the TurtleBot 3 Burger and ROS Kinetic. Master's thesis, University of Applied Sciences Vorarlberg (2020), https://opus.fhv.at/frontdoor/deliver/index/docId/3888/file/Gross_Daniel-Robot_Navigation_using_ROS.pdf

[22] Havelund, K., Peled, D.: An extension of ltl with rules and its application to runtime verification. In: Runtime Verification: 19th International Conference, RV 2019, Porto, Portugal, October 8–11, 2019, Proceedings 19. pp. 239–255. Springer (2019)

[23] Huang, C., Fan, J., Chen, X., Li, W., Zhu, Q.: Polar: A polynomial arithmetic framework for verifying neural-network controlled systems. In: International Symposium on Automated Technology for Verification and Analysis. pp. 414–430. Springer (2022)

[24] Huang, C., Fan, J., Li, W., Chen, X., Zhu, Q.: Reachnn: Reachability analysis of neural-network controlled systems. ACM Transactions on Embedded Computing Systems (TECS) **18**(5s), 1–22 (2019)

[25] Huang, J., Erdogan, C., Zhang, Y., Moore, B., Luo, Q., Sundaresan, A., Rosu, G.: Rosrv: Runtime verification for robots. In: Runtime Verification: 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings 5. pp. 247–254. Springer (2014)

[26] Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30. pp. 3–29. Springer (2017)

[27] Ivanov, R., Carpenter, T., Weimer, J., Alur, R., Pappas, G., Lee, I.: Verisig 2.0: Verification of neural network controllers using taylor model precondi-

tioning. In: International Conference on Computer Aided Verification. pp. 249–262. Springer (2021)
[28] Ivanov, R., Carpenter, T.J., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Case study: verifying the safety of an autonomous racing car with a neural network controller. In: Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control. pp. 1–7 (2020)
[29] Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 169–178 (2019)
[30] Jakšić, S., Bartocci, E., Grosu, R., Ničković, D.: An algebraic framework for runtime verification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **37**(11), 2233–2243 (2018)
[31] Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30. pp. 97–117. Springer (2017)
[32] Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al.: The marabou framework for verification and analysis of deep neural networks. In: Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31. pp. 443–452. Springer (2019)
[33] Kochdumper, N., Althoff, M.: Constrained polynomial zonotopes. Acta Informatica **60**(3), 279–316 (2023)
[34] Lee, E.A., Seshia, S.A.: Introduction to embedded systems: A cyber-physical systems approach. MIT press (2016)
[35] Leucker, M., Schmitz, M., à Tellinghusen, D.: Runtime verification for interconnected medical devices. In: International Symposium on Leveraging Applications of Formal Methods. pp. 380–387. Springer (2016)
[36] Li, J., Sun, J., Chen, G.: A multi-switching tracking control scheme for autonomous mobile robot in unknown obstacle environments. Electronics **9**(1) (2020). https://doi.org/10.3390/electronics9010042, https://www.mdpi.com/2079-9292/9/1/42
[37] Lindemann, L., Qin, X., Deshmukh, J.V., Pappas, G.J.: Conformal prediction for stl runtime verification. In: Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023). pp. 142–153 (2023)
[38] Liu, E.I., Althoff, M.: Computing specification-compliant reachable sets for motion planning of automated vehicles. In: 2021 IEEE Intelligent Vehicles Symposium (IV). pp. 1037–1044 (2021). https://doi.org/10.1109/IV48863.2021.9575739
[39] Liu, X., Huang, C., Wang, Y., Zheng, B., Zhu, Q.: Physics-aware safety-assured design of hierarchical neural network based planner. In: Cyber-Physical Systems (ICCPS), 2022 ACM/IEEE International Conference on (May 2022)

[40] Lopez, D.M., Choi, S.W., Tran, H.D., Johnson, T.T.: Nnv 2.0: the neural network verification tool. In: International Conference on Computer Aided Verification. pp. 397–412. Springer (2023)
[41] Qian, Z., Zhong, S., Sun, G., Xing, X., Jin, Y.: A formal approach to design and security verification of operating systems for intelligent transportation systems based on object model. IEEE Transactions on Intelligent Transportation Systems (2022)
[42] Sastry, S.: Nonlinear systems: analysis, stability, and control, vol. 10. Springer Science & Business Media (2013)
[43] Siwek, M., Panasiuk, J., Baranowski, L., Kaczmarek, W., Prusaczyk, P., Borys, S.: Identification of differential drive robot dynamic model parameters. Materials (Basel) **16**(2), 683 (2023). https://doi.org/10.3390/ma16020683
[44] Su, H., Feng, S., Zhan, S., Zhan, N.: Switching controller synthesis for hybrid systems against stl formulas. arXiv preprint arXiv:2406.16588 (2024)
[45] Tabuada, P.: Verification and Control of Hybrid Systems: A Symbolic Approach. Springer (2009)
[46] Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.J., Kolter, J.Z.: Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. Advances in Neural Information Processing Systems **34**, 29909–29921 (2021)
[47] Wang, Y., Huang, C., Wang, Z., Wang, Z., Zhu, Q.: Design-while-verify: correct-by-construction control learning with verification in the loop. In: Proceedings of the 59th ACM/IEEE Design Automation Conference. p. 925–930. DAC '22, Association for Computing Machinery, New York, NY, USA (2022), https://doi.org/10.1145/3489517.3530556
[48] Wang, Y., Zhan, S., Wang, Z., Huang, C., Wang, Z., Yang, Z., Zhu, Q.: Joint differentiable optimization and verification for certified reinforcement learning. In: Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023). pp. 132–141 (2023)
[49] Wang, Y., Zhan, S.S., Jiao, R., Wang, Z., Jin, W., Yang, Z., Wang, Z., Huang, C., Zhu, Q.: Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In: International Conference on Machine Learning. pp. 36593–36604. PMLR (2023)
[50] Wang, Y., Zhou, W., Fan, J., Wang, Z., Li, J., Chen, X., Huang, C., Li, W., Zhu, Q.: Polar-express: Efficient and precise formal reachability analysis of neural-network controlled systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2023)
[51] Wang, Z., Huang, C., Wang, Y., Hobbs, C., Chakraborty, S., Zhu, Q.: Bounding perception neural network uncertainty for safe control of autonomous systems. In: DATE'21: Proceedings of the Conference on Design, Automation and Test in Europe (2021)
[52] Wang, Z., Huang, C., Zhu, Q.: Efficient global robustness certification of neural networks via interleaving twin-network encoding. In: DATE'22: Proceedings of the Conference on Design, Automation and Test in Europe (2022)

[53] Wei, T., Wang, Y., Zhu, Q.: Deep reinforcement learning for building hvac control. In: 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC). pp. 1–6 (June 2017). https://doi.org/10.1145/3061639.3062224

[54] Wollschlaeger, M., Sauter, T., Jasperneite, J.: The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. IEEE industrial electronics magazine **11**(1), 17–27 (2017)

[55] Xiong, G., Zhu, F., Liu, X., Dong, X., Huang, W., Chen, S., Zhao, K.: Cyber-physical-social system in intelligent transportation. IEEE/CAA Journal of Automatica Sinica **2**(3), 320–333 (2015)

[56] Xu, S., Fu, Y., Wang, Y., Yang, Z., O'Neill, Z., Wang, Z., Zhu, Q.: Accelerate online reinforcement learning for building hvac control with heterogeneous expert guidances. In: Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation. p. 89–98. BuildSys '22, Association for Computing Machinery, New York, NY, USA (2022), https://doi.org/10.1145/3563357.3564064

[57] Xu, S., Wang, Y., Wang, Y., O'Neill, Z., Zhu, Q.: One for many: Transfer learning for building hvac control. In: Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation. p. 230–239. BuildSys '20, Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3408308.3427617

[58] Xue, B., Alba, C., Abraham, J., Kannampallil, T., Lu, C.: Prescribing large language models for perioperative care: What's the right dose for pre-trained models? arXiv preprint arXiv:2402.17493 (2024)

[59] Xue, B., Said, A.S., Xu, Z., Liu, H., Shah, N., Yang, H., Payne, P., Lu, C.: Assisting clinical decisions for scarcely available treatment via disentangled latent representation. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 5360–5371 (2023)

[60] Zapridou, E., Bartocci, E., Katsaros, P.: Runtime verification of autonomous driving systems in carla. In: International Conference on Runtime Verification. pp. 172–183. Springer (2020)

[61] Zhan, S.S., Wang, Y., Wu, Q., Jiao, R., Huang, C., Zhu, Q.: State-wise safe reinforcement learning with pixel observations. In: 6th Annual Learning for Dynamics and Control Conference (2024)

[62] Zhang, Z., An, J., Arcaini, P., Hasuo, I.: Online causation monitoring of signal temporal logic. In: International Conference on Computer Aided Verification. pp. 62–84. Springer (2023)

[63] Zhou, K., Doyle, J.C.: Robust and Optimal Control. Prentice Hall (1996)

[64] Zhu, Q., Huang, C., Jiao, R., Lan, S., Liang, H., Liu, X., Wang, Y., Wang, Z., Xu, S.: Safety-assured design and adaptation of learning-enabled autonomous systems. In: Proceedings of the 26th Asia and South Pacific Design Automation Conference. p. 753–760. ASPDAC '21, Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3394885.3431623

[65] Zhu, Q., Li, W., Huang, C., Chen, X., Zhou, W., Wang, Y., Li, J., Fu, F.: Verification and design of robust and safe neural network-enabled autonomous systems. In: 2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton). pp. 1–8. IEEE (2023)

[66] Zhu, Q., Li, W., Kim, H., Xiang, Y., Wardega, K., Wang, Z., Wang, Y., Liang, H., Huang, C., Fan, J., Choi, H.: Know the unknowns: Addressing disturbances and uncertainties in autonomous systems. In: Proceedings of the 39th International Conference on Computer-Aided Design. ICCAD '20, Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3400302.3415768

A Turtlebot Specification

To emulate real robot operations, we designed a robot testbed using the Flatland simulation environment. This setup replicates the dynamics of the Turtlebot 3 Burger, a differential wheeled robot equipped with two independently driven wheels and a LiDAR sensor [21]. Its maximum translational and angular velocities are 0.22 m/s and 2.84 rad/s, respectively. It has a 360-degree Laser Distance Sensor (LDS-01) capable of scanning the environment at 300 rpm, with a distance range of 120 mm to 3600 mm and a sample rate of 1.8k Hz. Given the Turtlebot's LiDAR scanning distance range, we set up a simulation with 5-meter bounded walls (Fig. 1) to ensure the robot receives appropriate laser scan values for localization.